



EXploitation des BDD.

LES DATAVIZ(s)





SQL Avancé



[instructions SQL 1]

Union

UNION

[instructions SQL 2];

L'objectif de la commande UNION de SQL est de combiner ensemble les résultats de deux requêtes.

Une restriction de UNION est que toutes les colonnes correspondantes doivent inclure le même type de données.

```
1  select pro_nom from proprietaire
2  Union
3  select voi_marque from voiture;
4
```

Attention, un union ne reprend pas les doublons = disctint. Il faut rajouter ALL pour obtenir les doublons :

```
select pro_nom from proprietaire
Union all
select voi_marque from voiture;
```







[instructions SQL 1]

INTERSECT

INTERSECT

[instructions SQL 2];

INTERSECT opère aussi sur deux instructions SQL. La différence entre les deux commandes est la suivante : UNION agit essentiellement comme un opérateur OR (OU) (la valeur est sélectionnée si elle apparaît dans la première ou la deuxième instruction)

la commande INTERSECT agit comme un opérateur AND (ET) operator (la valeur est sélectionnée seulement si elle apparaît dans les deux instructions).

Attention toutes colonnes correspondantes doivent inclure le même type de données.

```
police=# insert into proprietaire values (589, 'renault', 'david', 'calais');

INSERT 0 1

police=# select pro_nom from proprietaire
intersect
intersect
select voi_marque from voiture;

select voi_marque from voiture;

renault
(1 ligne)
```





MINUS / EXCEPT

[instructions SQL 1]

JINUS

[instructions SQL 2];

MINUS opère sur deux instructions SQL. Elle prend tous les résultats de la première instruction SQL, puis soustrait ceux de la deuxième instruction SQL pour obtenir la réponse finale.

Ex : Retrouvez les noms des propriétaires de voitures qui habitent Calais et qui ne sont pas propriétaires à Dunkerque (hypothèse ou ils ont 2 résidences)

```
select pro_nom from proprietaire where lower(pro_ville) like 'calais'

select pro_nom from proprietaire where lower(pro_ville) like 'dunkerque';

police=# select pro_nom from proprietaire where lower(pro_ville) like 'calais'
police-# EXCEPT
police-# select pro_nom from proprietaire where lower(pro_ville) like 'dunkerque'
police-#;
pro_nom

Higa
Terre
Roussy
renault
synave
(5 lignes)
```





EXISTS

Analyse de la requête below :

```
select nometu from etudiants et inner join avoir_note a on a.numetu=et.numetu
where a.numepr=2 and a.note >10;
```

La requête above répond à la question de savoir qui sont les noms des étudiants qui ont une note >10 à l'épreuve 2





EXISTS

EXISTS retourne tuples pour lesquels la condition est vraie

```
select nometu from etudiants where exists (select 1 from avoir_note where avoir_note.numetu=etudiants.

numetu and avoir_note.numepr=2 and avoir_note.note >10);

iut=# select nometu from etudiants where exists (select 1 from avoir_note where avoir_note.numetu=etudiants.numetu and avoir_note.numepr=2 and avoir_note.note >10);

roblin athur bagnole vendraux vendermaele marke (6 lignes)
```

Si la sous requête renvoie vrai alors on affiche les tuples correspondants à vrai Cad les noms des étudiants qui répondent vrai à la condition Exists





NOT EXISTS

EXISTS retourne tuples pour lesquels la condition est vraie. Il est donc possible de répondre par 3 façons à la requête ci-dessous :

« Donnez les noms des clients qui n'ont pas passé de commandes. »

```
entreprise=# select * from client;
numerocli | nomcli | prenomcli | datenaicli | villecli

1 | dupont | henry | 1995-10-10 | CALAIS
2 | durant | thierry | 1995-10-24 | CALAIS
3 | durant | ophélie | 1995-10-31 | CALAIS
(3 lignes)
```

```
TABLE(=relation) CLIENT
```

```
entreprise=# select * from commande;

numerocli | numeroprod

1 | 235

1 | 236

2 | 236

1 | 237

(4 lignes)
```

TABLE COMMANDE





EXISTS

« Donnez les noms des clients qui n'ont pas passé de commandes. »

<u>1ère possibilité avec « NOT IN »</u>

- select numerocli, nomcli from client where numerocli
- NOT IN (select numerocli from commande);

```
entreprise=# select * from client;
numerocli | nomcli | prenomcli | datenaicli | villecli

1 | dupont | henry | 1995-10-10 | CALAIS
2 | durant | thierry | 1995-10-24 | CALAIS
3 | durant | ophélie | 1995-10-31 | CALAIS

(3 lignes)

numerocli

nomcli

durant
```





NOT EXISTS

« Donnez les noms des clients qui n'ont pas passé de commandes. »

<u>2ème possibilité avec «NOT EXISTS »</u>

```
select numerocli, nomcli from client as cl where not exists (select numerocli from commande as c where c.numerocli=cl.numerocli);
```

durant

```
entreprise=# select * from client ;
numerocli | nomcli | prenomcli | datenaicli | villecli

1 | dupont | henry | 1995-10-10 | CALAIS
2 | durant | thierry | 1995-10-24 | CALAIS
3 | durant | ophélie | 1995-10-31 | CALAIS
(3 lignes)

numerocli

nomcli
```

Le client 3 renvoie vrai À la sous requête Not Exists

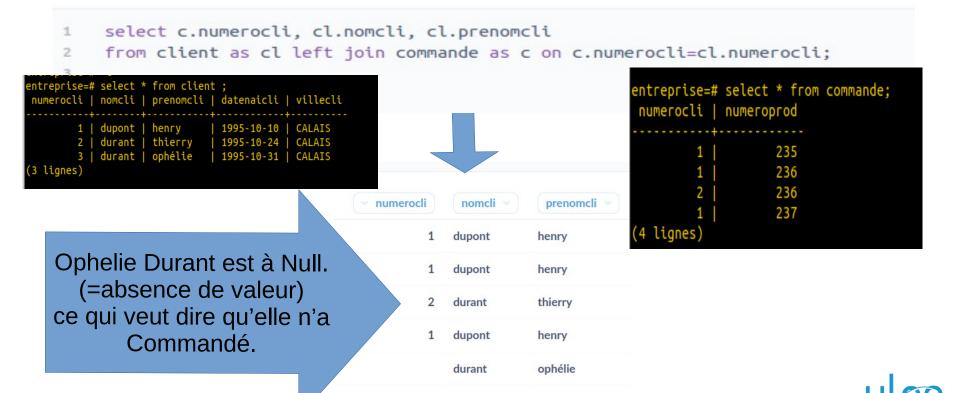




EXISTS

« Donnez les noms des clients qui n'ont pas passé de commandes. »

<u>3ème possibilité avec « LEFT JOIN »</u>





EXISTS

« Donnez les noms des clients qui n'ont pas passé de commandes. »

<u>3ème possibilité avec « LEFT JOIN »</u>

n'a pas commandé

select c.numerocli, cl.nomcli, cl.prenomcli from client as cl left join commande as c on c.numerocli=cl.numerocli where c.numerocli is null; treprise=# select * from client ; entreprise=# select * from commande; merocli | nomcli | prenomcli | datenaicli | villecli numerocli | numeroprod CALAIS 235 durant | ophélie 1995-10-31 | CALAIS 236 237 (4 lignes) En rajoutant le filtre WHERE avec IS NULL prenomcli numerocli nomcli Sur numerocli, on récupère Le client qui durant ophélie





NOT EXISTS (mode division)

La division d'une table t1 (dividende) par la table t2 (diviseur) correspond à une table t3 (quotient) composés de tuples.

Donc la multiplication de t3 par t2 donne un extrait de t1.

designation	taille
+	+
lacoste	S
lacoste	m
lacoste	l l
lacoste	xl
chanel	S
chanel	m
tommy	S
tommy	m
tommy	l
tommy	xl
hugo	S
hugo	m
hugo	l



Lacoste chanel





NOT EXISTS (mode division)

La division d'une table t1 (dividende) par la table t2 (diviseur) correspond à Une table t3 (quotient) composés de tuples.

Donc la multiplication de t3 par t2 donne un extrait de t1.

QUESTION : quels sont les marques de polo qui ont toutes les tailles ?









NOT EXISTS (mode division)

QUESTION : quels sont les polos qui ont toutes les tailles (table taille) dans la table polo ?

designation	taille
+	+
lacoste	S
lacoste	m
lacoste	l l
lacoste	xl
chanel	S
chanel	m
tommy	S
tommy	m
tommy	l
tommy	xl
hugo	S
hugo	m
hugo	l



Lacoste chanel





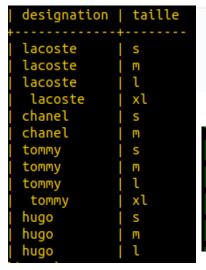
NOT EXISTS (division) technique:

QUESTION: quels sont les polos qui ont toutes les tailles (table taille) dans la table polo?

La condition est vraie lorsque la sous requête ne donne pas de résultat

- 1. Extraire les polos pour lesquels il y a des tailles manquantes
- 2. Ne pas sélectionner les polos ayant des tailles manquantes

Pour comprendre : on va essayer de trouver les tailles manquantes de la marque CHANEL (1er essai) dont on sait qu'il y a des tailles manquantes



```
select * from taille t where not exists
(select * from polo p where marque like 'chanel' and p.taille=t.taille);
          id
                                 designation
                   taille
                                large
                                extra large
```







NOT EXISTS (division) technique:

QUESTION: quels sont les polos qui ont toutes les tailles (table taille) dans la table polo?

La condition est vraie lorsque la sous requête ne donne pas de résultat

- 1. extraire les polos pour lesquels il y a des tailles manquantes
- 2. Ne pas sélectionner les polos ayant des tailles manquantes

Pour comprendre : on va essayer de trouver les tailles manquantes de la marque lacoste qui existe dans toutes les tailles (2ème essai).



La marque lacoste possédant toutes les tailles correspond bien à la sous requête demandée Cad de posseder toutes les tailles . Donc elle n'apparait pas car ne correspond a not exists !!





NOT EXISTS (division) technique:

Pour comprendre : on va essayer de trouver les tailles appartenant à lacoste cette fois-ci en mettant exists.



La marque lacoste possédant toutes les tailles correspond bien à la sous requête demandée Cad de posseder toutes les tailles . Donc elle renvoie toutes les tailles de lacoste !!





NOT EXISTS (division)

```
-- on souhaite les marques qui possèdent toutes les tailles
-- donc on va chercher ces marques de polo dans une première table polo

select distinct(p1.marque) from polo p1 where not exists
-- puis on enlève les marques qui ne possèdent pas toutes les tailles
-- en remplacant la marque du polo par son nom de colonne
(select * from taille t where not exists
(select * from polo p2 where p2.marque=p1.marque and p2.taille=t.taille));
```



On remplace le nom de la marque du polo par le nom de sa colonne et on duplique la table polo pour comparer.



On obtient bien les 2 marques qui possèdent toutes les tailles après les avoir enlever des autres polos qui ne possèdent pas toutes les marque.





Division : 2ème méthode classique

select p.marque, count(distinct(t.taille)) from polo as p inner join taille t on t.taille=p.taille group by p.marque;



1. En premier, je cherche le nombre de tailles par marque





Division: 2ème méthode

2. E deuxième, je cherche le nombre de tailles au total dans la table taille

```
select count()*) from taille;
```

3. Je compare avec la requête précédente à savoir de trouver les marques de polo qui ont 4 tailles!

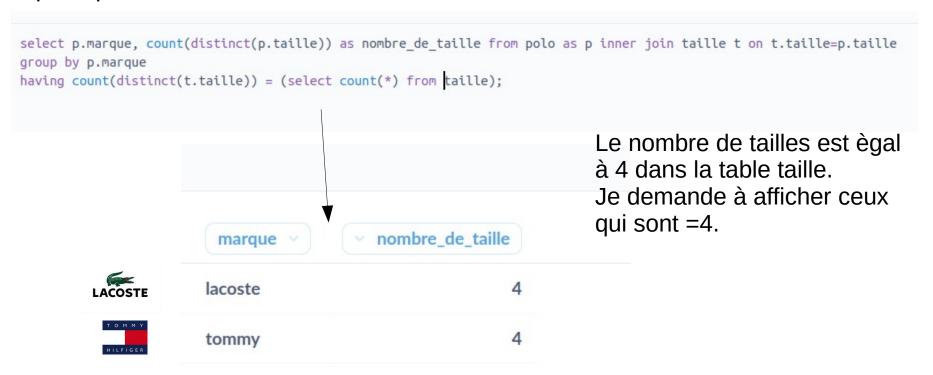
```
select p.marque, count(distinct(p.taille)) as nombre_de_taille from polo as p inner join taille t on t.taille=p.taille
group by p.marque
having count(distinct(t.taille)) = (select count(*) from taille);
```





Division: 2ème méthode

3. Je compare avec la requête précédente à savoir de trouver les marques de polo qui ont 4 tailles !







Un WHERE après un GROUP BY possible ??

```
select a.numetu , avg(note) as moyenne from etudiants e inner join avoir_note a on e.numetu=a.numetu group by 1
order by moyenne ASC;
```

order by avg(note) ASC;		
numetu		
3	5	
7	9	
13	11	
11	11	
8	12	
18	12.5457022984823	
17	13.8	
5	14	
12	14.5	
2	18	
15	18	
10	18	
9	18	
14	18.5	
1	18.5	
6	19	
16	19	
4	19	
(18 ligne	es)	

On souhaite garder uniquement les numetu qui ont une moyenne >=15.

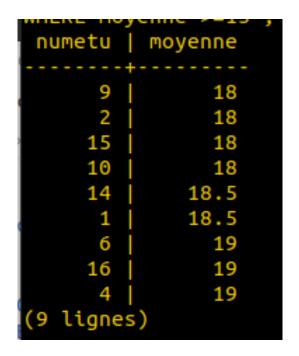
Il est possible de transformer la requête en table !!





Un WHERE après un GROUP BY possible ??

```
select * from
(select a.numetu , avg(note) as moyenne from etudiants e inner join avoir_note a on e.numetu=a.numetu
group by 1
order by moyenne ASC)
AS moyenne_par_etudiant
WHERE moyenne >=15 ;
```



On souhaite grader uniquement les numetu qui ont une moyenne >=15.

On renomme la requête en table moyenne par etudiant.

Puis on fait un where !!





LES BOOLEENS.

Créer une condition qui vérifie Vraie ou Faux ou VIDE

```
Select
numetu,
prenometu,
prenometu = 'lea' as prenomlea
from etudiants;
```

On met la conditon = 'lea' en troisième . Tous les prenoms sont checkés pour savoir si lea = vrai (t) ou faux (f)

```
prenometu | prenomlea
numetu
          lea
          leon
          luc
          sophie
          marc
         marc
         helene
          loic
          godard
         marie
    10
         elsa
    11
         elise
          pierre
    13
          luc
          jules
          luc
    16
          loic
    17
          leon
         prenom
(19 lignes)
```





CASE .. WHEN

Permet d'exprimer des conditions sur une requête

```
CASE
WHEN condition THEN result
[WHEN ...]
[WHEN ...]
[ELSE result]
END
```





CASE .. WHEN

Faire un count sur la villetudiant en distinguant les calaisiens et les autres

```
Select

Case · When · viletu · like · 'calais' · THEN · 'calaisiens' · ELSE · 'autre_villes'

END · AS · calaisiens_et_les_non_calaisiens,

Count(*) · as · nombre_etudiant

from · etudiants

group · by · calaisiens_et_les_non_calaisiens;

On a renommé le case

'calaisiens et les non calaisiens

pour ensuite l'utiliser dans un

group by !!!
```

```
calaisiens_et_les_non_calaisiens | nombre_etudiant

autre_villes | 13

calaisiens | 6

(2 lignes)
```





CASE .. WHEN OU FILTER

FILTERING enables you to filter an aggregate function by conditional expressions.

```
SELECT
    COUNT(CASE WHEN condition THEN 1 END) AS column_name_1,
    COUNT(*) FILTER (WHERE condition) AS column_name_2,
FROM table
```

Il est possible de filtrer graçe à la commande FILTER





CASE .. WHEN OU FILTER

Autre façon de répondre à la question du nombre de calaisiens ou pas avec un CASE ou bien un FILTER

```
Select
count (CASE when viletu like 'calais' THEN 1 ELSE NULL END) AS nombre_de_calaisiens,
count(*) FILTER (WHERE viletu NOT LIKE 'calais') AS nonbre_de_non_calasiens
from etudiants;
```

```
iut=# Select
iut-# count (CASE when viletu like 'calais' THEN 1 ELSE NULL END) AS nombre_de_calaisiens,
iut-# count(*) FILTER ( WHERE viletu NOT LIKE 'calais') AS nonbre_de_non_calasiens
iut-# from etudiants;
nombre_de_calaisiens | nonbre_de_non_calasiens
6 | 13
(1 ligne)
```







Mediane et quartiles

On utilise la fonction PERCENTILE

On souhaite connaître la médiane par étudiant

Il faut rentrer la valeur 0,5 à la fonction percentile_disc si on veut la médiane. 0,25 pour le premier quart Oui 0,75 pour le deuxième

```
19
11
13
15
```

```
numetu,
percentile_disc(0.5) within group (order by note) as "mediane"
from avoir_note
group by numetu;
```







Mediane et quartiles

On utilise la fonction PERCENTILE

On souhaite connaître la médiane par étudiant

Il faut rentrer la valeur 0,5 à la fonction percentile_disc si on veut la médiane. 0,25 pour le premier quart Oui 0,75 pour le deuxième

```
19
11
13
15
```

```
numetu,
percentile_disc(0.5) within group (order by note) as "mediane"
from avoir_note
group by numetu;
```

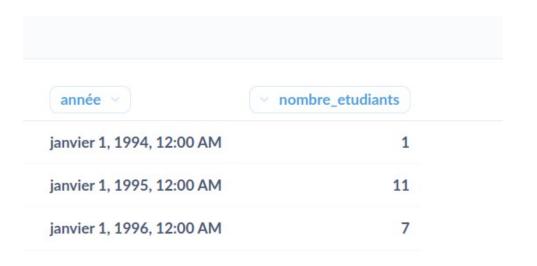




Date_Trunc

On utilise la Date_Trunc pour regrouper des Calculs par « groupe de temps »

Ex : on souhaite connaître le nombre d'étudiant total par année de naissance





Nouvelle question

```
iut ~

select
date_trunc('year',datentetu) AS Année,
count(*) as nombre_etudiants
From etudiants
group by Année
order by Année;
```

```
select
date_trunc('year',datentetu) AS Année,
count(*) as nombre_etudiants
From etudiants
group by 1
order by 1;
```





Date_part

On utilise la Date_part pour utiliser le numéro de mois ou de semaine pour grouper des groupes

Ex : on souhaite connaître le nombre d'étudiant total par mois de naissance. (avec le chiffre du mois)

```
iut ~

select
date_part('month',datenaietu) AS Mois,
count(*) as nombre_etudiant
From etudiants
group by 1
order by 1;
```

v mois	v nombre_etudiant
1	4
2	1
3	4
4	4
5	2
6	1
7	1
11	2





Calculer la différence entre 2 dates et concaténer 2 strings

Ex : on souhaite l'age des étudiants en années. (sans utiliser la fonction AGE)

```
select
1
     concat(nometu, ' ', prenometu) as nom_complet,
     datenaietu as date de naissance,
3
     now() as date_courante,
4
     date_part ('day', now() - datenaietu)/365 as age
5
     from etudiants:
6
                     date_de_naissance
                                         date_courante
    nom_complet
                                                            age
   roblin lea
                    14 janvier, 1975
                                        8 mai, 2022
                                                           47.35
   athur leon
                    12 avril, 1974
                                        8 mai, 2022
                                                            48.1
   minol luc
                    12 mars, 1977
                                        8 mai, 2022
                                                           45.19
```





LES JOINTURES

LEFT JOIN

All customers, regardless if they have transactions or not

customer_id	transaction_id
id_1	transc_id_1
id_2	NULL
id_3	transac_id_2
id_4	NULL

INNER JOIN

```
SELECT *
FROM customers AS c
INNER JOIN transactions AS t
ON t.customer_id = c.id
```

All customers that have at least one transaction

transaction_id
transc_id_1
transac_id_2

RIGHT JOIN

```
SELECT *
FROM customers AS c
RIGHT JOIN transactions AS t
    ON t.customer_id = c.id
```

All transactions, regardless if they have a customer or not

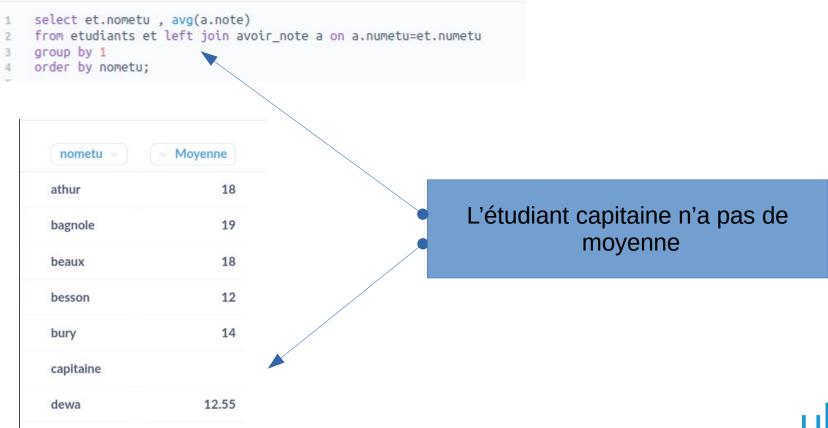
customer_id	transaction_id
id_1	transc_id_1
id_3	transac_id_2
NULL	transac_id_3





LEFT JOIN

Ex : Existe-t-il des étudiants qui n'ont pas de moyenne ?

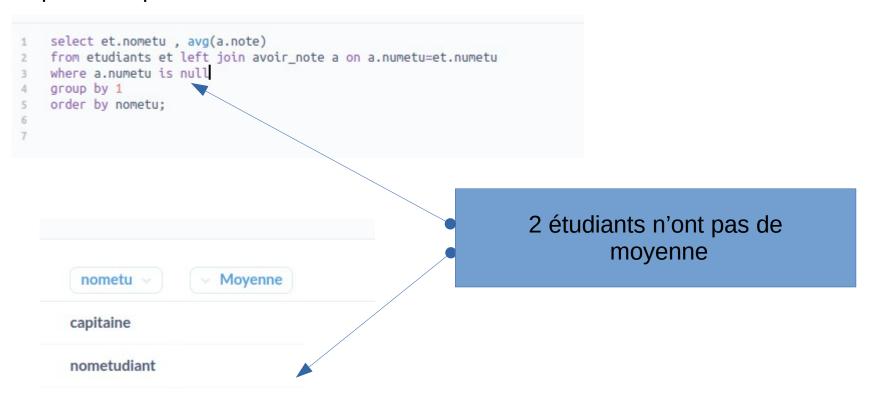






LEFT JOIN

Ex : Existe-t-il des étudiants qui n'ont pas de moyenne ? Avec le where pour récupérer ces étudiants.

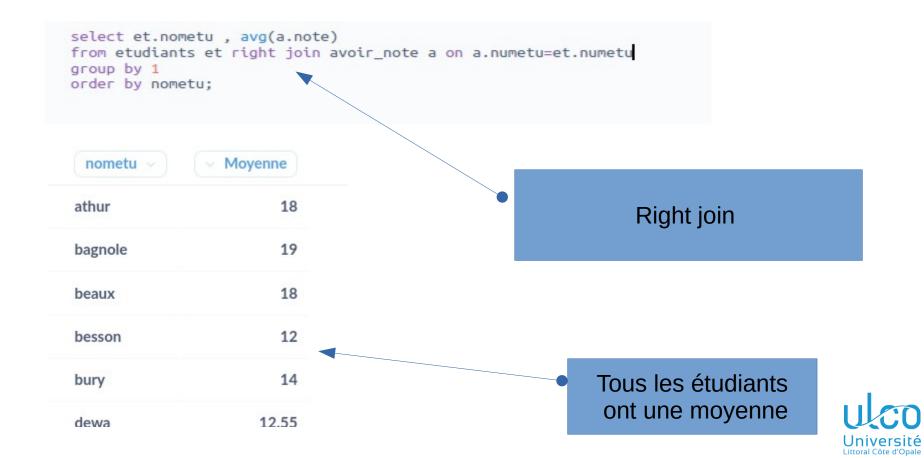






RIGHT JOIN

Ex : Existe-t-il des moyennes d'étudiants qui ne sont pas rattachés à un étudiant ?





CTE: Common Table Expression. WITH + UNE CLAUSE (time_series)

Il est possible de décomposer sa requête complexe en plusieurs petites tables qu'on peut définir soi même. (Pour s'en resservir).

WITH clause is used to transform complex queries syntax into simplest tables you can reuse then.

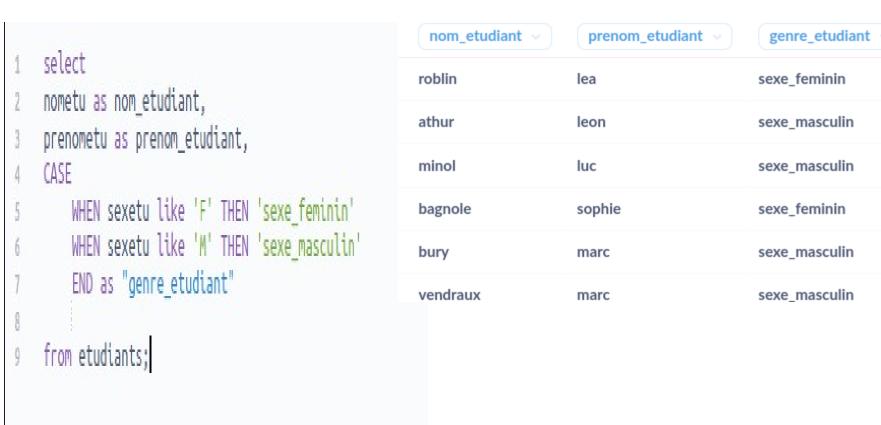
```
WITH time_series AS (
    SELECT
                                                                     Create time series table
        months
                                                                     using generate_series syntax
    FROM generate_series('2020-01-01'::timestamp,
        '2020-05-01'::timestamp, '1 month') AS months
,customer_creation AS (
                                                                     Create customer creation
        DATE_TRUNC('month', created_at)::date AS month
                                                                     table using the table
    FROM transactions
                                                                     transactions
SELECT
    t.months AS series_month,
    c.month AS transaction_month
FROM time_series AS t
                                                                     REUSE THESE 2 TABLES. YOU
INNER JOIN customer_creation AS c
                                                                     CAN JOIN THEM, OR USE
    ON c.month = t.months
                                                                    THEM ALONE
LIMIT 1
```





CTE: Common Table Expression. WITH CLAUSE

Exemple: 1. Affichons la table des étudiants avec nom, prenom et genre







CTE: Common Table Expression. WITH CLAUSE

Exemple : 2. Création d'une CTE cad une nouvelle table qui va s'appeler type_etudiant avec un WITH







CTE: Common Table Expression. WITH CLAUSE

Exemple : 3. Création d'une troisième table CTE qui reprend les notes >10 Et ensuite requête sur ces 2 nouvelles tables CTE

```
prenometu
                                                                                                                   note
                                                                                                                       20
                                                                                           luc
                                                                                           elise
                                                                                                                       20
    -- je rajoute un with puis j'ajoute le nom de la nouvelle table
    with genre_etudiant AS (
    select
                                                                                           loic
                                                                                                                       19
    et.numetu .
    et.nometu .
    et.prenometu ,
                                                                                           loic
                                                                                                                       13
    CASE
        WHEN sexetu like 'F' THEN 'sexe_feminin'
        WHEN sexetu like 'M' THEN 'sexe_masculin'
                                                                                           loic
                                                                                                                       11
        END as "genre etudiant"
11
                                                                                                                       19
    from etudiants as et)
12
                                                                                           marc
13
    -- creation de la deuxième table (CTE) qui reprend les notes supérieures à 10
14
15
    , note_superieures_10 as (select A.numetu, A.note from avoir_note AS A where A.note>10)
16
17
18
    -- je fais une requête sur ces 2 nouvelles tables qui me donne les noms des etudiants qui ont une note >10
19
    select ge.prenometu, n.note from genre etudiant as ge inner join note superieures 10 as n on ge.numetu=n.numetu;
```





Somme cumulative. (Peut servir pour dataviz)

```
business_intelligence_courses 

WITH daily_transactions AS (
SELECT
DATE_TRUNC('day', created_at) AS day,
COUNT(*) AS nb_transactions
FROM transactions
GROUP BY 1

SELECT
day,
SUM(nb_transactions) OVER (ORDER BY day ASC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_transactions --- cumulative data over days
FROM daily_transactions
```

Requête qui donne le cumul des ventes jour après jour

day	cumulative_transactions
February 6, 2020, 12:00 AM	1
February 7, 2020, 12:00 AM	3
February 14, 2020, 12:00 AM	5
February 15, 2020, 12:00 AM	9
February 16, 2020, 12:00 AM	18
February 19, 2020, 12:00 AM	34





Sauvegarde d'une BDD

PG_DUMP

```
important mais cela reste une image

complémentaire à l'archive log

pas de méthode unique => à adapter à sa situation

ne copie pas les users et autres spécifiques à l'instance

deux outils (complémentaires) :
    +- pg_dump : personnalisation au maximum
    +- pg_dumpall : backup intégral de l'instance (y compris users...)
```





Sauvegarde d'une BDD

PG_DUMP

```
Beaucoup d'options (une partie identique à psql) :
 +- -h : hostname (eh oui c'est pas l'aide)
 +- -d : database
 +- -p : port
 +- -U : user
 +- -f : fichier de sortie
 +- -F : précision du format de sortie
     +- c : custom (binaire)
     +- d : directory (permet la parallélisation)
        t : tar
     +- p : plain text
 +- -j : parallélisation
```





Sauvegarde d'une BDD

PG_DUMP:

```
PostgreSQL database dump
- Dumped from database version 9.5.25
- Dumped by pg dump version 9.5.25
SET statement timeout = 0;
SET lock timeout = 0;
SET client encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check function bodies = false;
SET xmloption = content;
SET client min messages = warning;
SET row security = off;
```

```
dany@dany-HP-ZBook-15-G2:~$ pg_dump -C mabase
```





Restauration d'une BDD

Psql -f : pour les fichiers sql (palin text) et pg_restore pour les fichiers binaires

```
+- sauvegarde d'une database :

pg_dump -d madatabase > madatabase.sql

+- restauration

dropdb madatabase
createdb madatabase && psql -f madatabase.sql

+- ou plus simplement si le dump prévoit le CREATE DATABA
SE -C

pg_dump -C -d madatabase > madatabase.sql
psql -F madatabase.sql
```

