

# 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
```



pro_nom
Vanda
citroen
Mercedes-Benz
jean
ferrari
Tesla
Musk
AIXAM
Fiat
Terre
Seidman
pedro
garen
BMW

Attention, un union ne reprend pas les doublons = distinct. 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=# select * from voiture;
police=# insert into proprietaire values (589, 'renault', 'david', 'calais');
INSERT 0 1
```

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

```
police=# select pro_nom from proprietaire
intersect
select voi_marque from voiture;
pro_nom
-----
renault
(1 ligne)
```

[instructions SQL 1]

# MINUS / EXCEPT

MINUS

[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)

```
1 select pro_nom from proprietaire where lower(pro_ville) like 'calais';
2 EXCEPT
3 select pro_nom from proprietaire where lower(pro_ville) like 'dunkerque';
4
```

```
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.numepre=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

```
iut=# select nometu from etudiants et inner join avoir_note a on a.numetu=et.numetu where a.numepre=2 and a.note >10;
      nometu
-----
marke
vendermaele
bagnole
roblin
athur
vendraux
(6 lignes)
```

# 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.numepre=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.numepre=2 and avoir_note.note >10);  
 nometu  
-----  
roblin  
athur  
bagnole  
vendreaux  
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. »

1ère possibilité avec « NOT IN »

```
1 select numerocli, nomcli from client where numerocli
2 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)
```



```
entreprise=# select * from commande;
 numerocli | numeroprod
-----+-----
          1 |         235
          1 |         236
          2 |         236
          1 |         237
(4 lignes)
```

▼ numerocli

nomcli ▼

3 durant

# NOT EXISTS

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

2ème possibilité avec «NOT EXISTS »

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

```
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 ▼

3      durant

Le client 3 renvoie vrai  
À la sous requête  
Not Exists

```
entreprise=# select * from commande;
 numerocli | numeroprod
-----+-----
      1 |      235
      1 |      236
      2 |      236
      1 |      237
(4 lignes)
```

# EXISTS

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

3ème possibilité avec « LEFT JOIN »

```
1 select c.numerocli, cl.nomcli, cl.prenomcli
2 from client as cl left join commande as c on c.numerocli=cl.numerocli;
```

```
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)
```



```
entreprise=# select * from commande;
numerocli | numeroprod
-----+-----
1 | 235
1 | 236
2 | 236
1 | 237
(4 lignes)
```

Ophelie Durant est à Null.  
(=absence de valeur)  
ce qui veut dire qu'elle n'a  
Commandé.

▼ numerocli	nomcli ▼	prenomcli ▼
1	dupont	henry
1	dupont	henry
2	durant	thierry
1	dupont	henry
	durant	ophélie

# EXISTS

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

3ème possibilité avec « LEFT JOIN »

```
1 select c.numerocli, cl.nomcli, cl.prenomcli
2 from client as cl left join commande as c on c.numerocli=cl.numerocli
3 where c.numerocli is null;
```

```
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)
```

```
entreprise=# select * from commande;
 numerocli | numeroprod
-----+-----
          1 |         235
          1 |         236
          2 |         236
          1 |         237
(4 lignes)
```

En rajoutant le filtre  
WHERE avec IS NULL  
Sur numerocli, on récupère  
Le client qui  
n'a pas commandé

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
lacoste	xl
chanel	s
chanel	m
tommy	s
tommy	m
tommy	l
tommy	xl
hugo	s
hugo	m
hugo	l

s
m
l
xl

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
lacoste	xl
chanel	s
chanel	m
tommy	s
tommy	m
tommy	l
tommy	xl
hugo	s
hugo	m
hugo	l

s
m
l
xl

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 (1<sup>er</sup> essai) dont on sait qu'il y a des tailles manquantes

designation	taille
lacoste	s
lacoste	m
lacoste	l
lacoste	xl
chanel	s
chanel	m
tommy	s
tommy	m
tommy	l
tommy	xl
hugo	s
hugo	m
hugo	l

s
m
l
xl

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

id	taille	designation
3	l	large
4	xl	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).

```
mabase=# select * from taille t where not exists
mabase-# (select * from polo p where marque like 'lacoste' and p.taille=t.taille);
 id | taille | designation
----+-----+-----
(0 ligne)
```



La marque lacoste possédant toutes les tailles correspond bien à la sous requête demandée Car de posséder toutes les tailles . Donc elle n'apparaît pas car ne correspond pas à not exists !!

# NOT EXISTS ( division) technique:

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

```
1 select * from taille t where exists
2 (select * from polo p where marque like 'lacoste' and p.taille=t.taille);
```

id	taille	description
1	s	petite taille
2	m	taille moyenne
3	l	large
4	xl	extra large



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

# NOT EXISTS (division)

```

1  -- on souhaite les marques qui possèdent toutes les tailles
2  -- donc on va chercher ces marques de polo dans une première table polo
3
4  select distinct(p1.marque) from polo p1 where not exists
5  -- puis on enlève les marques qui ne possèdent pas toutes les tailles
6  -- en remplaçant la marque du polo par son nom de colonne
7  (select * from taille t where not exists
8  (select * from polo p2 where p2.marque=p1.marque and p2.taille=t.taille));

```



marque ▾



lacoste



tommy

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;
```

marque ▾	▾ Nombre de lignes
chanel	2
hugo	3
lacoste	4
tommy	4

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

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

4

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 !

```
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);
```

Le nombre de tailles est égal à 4 dans la table taille.  
Je demande à afficher ceux qui sont =4.



	marque ▼	nombre_de_taille ▼
	lacoste	4
	tommy	4

## 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 |      avg
-----+-----
      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 lignes)
```

On souhaite garder uniquement les numetu qui ont une moyenne  $\geq 15$ .

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 ;
```

WHERE moyenne >=15 ;

numetu	moyenne
9	18
2	18
15	18
10	18
14	18.5
1	18.5
6	19
16	19
4	19

(9 lignes)

On souhaite grader uniquement les numetu qui ont une moyenne  $\geq 15$ .

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 condition = 'lea' en troisième.  
Tous les prénoms sont vérifiés pour savoir si lea = vrai (t) ou faux (f)

numetu	prenometu	prenomlea
1	lea	t
2	leon	f
3	luc	f
4	sophie	f
5	marc	f
6	marc	f
7	helene	f
8	loic	f
9	godard	f
10	marie	f
11	elsa	f
12	elise	f
13	pierre	f
14	luc	f
15	jules	f
16	luc	f
17	loic	f
18	leon	f
155	prenom	f

(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 grâce 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 nombre_de_non_calaisiens
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 nombre_de_non_calaisiens
iut-# from etudiants ;
  nombre_de_calaisiens | nombre_de_non_calaisiens
-----+-----
                        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

étu-#	group by numetu;
numetu	mediane
1	19
2	17
3	3
4	19
5	9
6	19
7	1
8	7
9	17
10	17
11	3
12	9
13	3
14	17
15	17
16	19
17	13

```
select
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

étu-#	group by numetu;
numetu	mediane
1	19
2	17
3	3
4	19
5	9
6	19
7	1
8	7
9	17
10	17
11	3
12	9
13	3
14	17
15	17
16	19
17	13

```
select
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

année ▾	▾ nombre_etudiants
janvier 1, 1994, 12:00 AM	1
janvier 1, 1995, 12:00 AM	11
janvier 1, 1996, 12:00 AM	7



## Nouvelle question

iut ▾

```

1  select
2  date_trunc('year',datentetu) AS Année,
3  count(*) as nombre_etudiants
4  From etudiants
5  group by Année
6  order by Année;

```

7  
8 |

```

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 ▾

```

1  select
2  date_part('month',datenaietu) AS Mois,
3  count(*) as nombre_etudiant
4  From etudiants
5  group by 1
6  order by 1;
7
8

```

mois	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'âge des étudiants en années. (sans utiliser la fonction AGE)

```

1  select
2  concat(nometu, ' ', prenometu) as nom_complet,
3  datenaietu as date_de_naissance,
4  now() as date_courante,
5  date_part('day', now() - datenaietu)/365 as age
6  from etudiants;

```

nom_complet ▾	date_de_naissance ▾	date_courante ▾	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

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

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

customer_id	transaction_id
id_1	transc_id_1
id_3	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 ?

```
1 select et.nometu , avg(a.note)
2 from etudiants et left join avoir_note a on a.nometu=et.nometu
3 group by 1
4 order by nometu;
```

nometu	Moyenne
athur	18
bagnole	19
beaux	18
besson	12
bury	14
capitaine	
dewa	12.55

L'étudiant capitaine n'a pas de moyenne

## LEFT JOIN

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

```

1  select et.nometu , avg(a.note)
2  from etudiants et left join avoir_note a on a.nometu=et.nometu
3  where a.nometu is null
4  group by 1
5  order by nometu;
6
7
    
```

	nometu	Moyenne
capitaine		
nometudiant		

2 étudiants n'ont pas de moyenne

## RIGHT JOIN

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

```
select et.nometu , avg(a.note)
from etudiants et right join avoir_note a on a.nometu=et.nometu
group by 1
order by nometu;
```

nometu	Moyenne
athur	18
bagnole	19
beaux	18
besson	12
bury	14
dewa	12.55

Right join

Tous les étudiants  
ont une moyenne

## 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
    months
  FROM generate_series('2020-01-01'::timestamp,
    '2020-05-01'::timestamp, '1 month') AS months
)

, customer_creation AS (
  SELECT
    DATE_TRUNC('month', created_at)::date AS month
  FROM transactions
)

SELECT
  t.months AS series_month,
  c.month AS transaction_month
FROM time_series AS t
INNER JOIN customer_creation AS c
  ON c.month = t.months
LIMIT 1
```

Create **time\_series** table  
using *generate\_series* syntax

Create **customer\_creation**  
table using the table  
*transactions*

**REUSE THESE 2 TABLES. YOU  
CAN JOIN THEM, OR USE  
THEM ALONE**

## CTE : Common Table Expression. WITH CLAUSE

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

```

1 select
2   nometu as nom_etudiant,
3   prenetu as prenom_etudiant,
4   CASE
5     WHEN sexetu like 'F' THEN 'sexe_feminin'
6     WHEN sexetu like 'M' THEN 'sexe_masculin'
7     END as "genre_etudiant"
8   .
9   from etudiants;

```

nom_etudiant	prenom_etudiant	genre_etudiant
roblin	lea	sexe_feminin
athur	leon	sexe_masculin
minol	luc	sexe_masculin
bagnole	sophie	sexe_feminin
bury	marc	sexe_masculin
vendraux	marc	sexe_masculin



## 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

```

1  -- je rajoute un with puis j'ajoute le nom de la nouvelle table
2  with genre_etudiant AS (
3  select
4  nometu as nom_etudiant,
5  prenometu as prenom_etudiant,
6  CASE
7    WHEN sexetu like 'F' THEN 'sexe_feminin'
8    WHEN sexetu like 'M' THEN 'sexe_masculin'
9    END as "genre_etudiant"
10  ...
11  from etudiants)
12
13
14  -- je fais une requête sur cette nouvelle table et j'obtiens un résultat identique
15  select * from genre_etudiant;
16

```

nom_etudiant	prenom_etudiant	genre_etudiant
roblin	lea	sexe_feminin
athur	leon	sexe_masculin
minol	luc	sexe_masculin
bagnole	sophie	sexe_feminin

## 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

```

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

```

prenometu	note
luc	20
elise	20
loic	19
loic	13
loic	11
marc	19

2 nouveaux attributs issus des 2 tables CTE

## Somme cumulative. (Peut servir pour dataviz)

business\_intelligence\_courses ▾

```

1 WITH daily_transactions AS (
2   SELECT
3     DATE_TRUNC('day', created_at) AS day,
4     COUNT(*) AS nb_transactions
5   FROM transactions
6   GROUP BY 1
7 )
8
9 SELECT
10   day,
11   SUM(nb_transactions) OVER (ORDER BY day ASC ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_transactions --- cumulative data over days
12 FROM daily_transactions
13
14
    
```

Création d'une CTE qui reprend le nombre de vente par jour

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 :

```
postgres@dany-HP-ZBook-15-G2:/home/dany$ pg_dump -C entreprise
--
-- 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 DATABASE -C
```

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

```
psql -F madatabase.sql
```