Sous requêtes

> *Principe: On utilise le résultat d'une requête dans une dans une autre requête*

1. Sous-requête dans le FROM

1.1. Exemple

On veux calculer le stock du magasin ayant le plus gros stock

> *Sous-requête : pour chaque magasin, son stock*

```
SELECT id_magasin, SUM(quantite) AS st
FROM stock
GROUP BY id_magasin;
```

id	st
1	150
2	193

`requête final`

```
SELECT MAX(st)
FROM (

  SELECT id_magasin, SUM(quantite) AS st
  FROM stock
  GROUP BY id_magasin
  ) AS S
```

`S` alias pour la sous requête

1.2. Remarque 1:

On est obligé de mettre un alias à la sous requête (uniquement sous-requête dans le `FROM`)

1.3. Remarque 2:

l'alias `st` du nom de colonne est utilisable uniquement à l'exterieur de la sous requete dans le from (sauf order by)

Autre syntaxe:

```
WITH s AS
( SELECT id_magasin,
  SUM(quantite) AS st
  FROM stock GROUP BY id_magasin)
  SELECT MAX(st)
  FROM s
```

1.4. Syntaxe du WITH:

```
  WITH R1 AS (requete\_1),
       R2 AS (requete\_2),
       ...
       Rn AS (requete\_n)
  SELECT (________); (requete principle)
```

dans requete_i, on peut utiliser R1,..., Ri-1, et bien sur dans la requete principale on peut utiliser R1,...,Rn

1.5. Autre exemple:

Produit qui ne sont pas dans tout les magasins, ie: les produits ayant la propriete : le produit n'est pas en stock dans au moins un magasin. Sous-requete couple (id_produit, id_magasin) où id_produit pas en stock dans id_magasin.

```
WITH s AS
(SELECT id_produit, id_magasin
FROM produit, magasin
EXCEPT
  SELECT id_produit, id_magasin
  FROM stock)
SELECT DISTINCT id_produit, desc_produit, prix
FROM produit
NATURAL JOIN s
```

1.6. Sous-requete dans le WHERE

2. 1) avec =, <=, >=, ...

`IS DISTINCT FROM` `IS NOT DISTINCT FROM`

2.1. Syntaxe:

Condition: valeur comparaison la sous-requete doit retourner 1 seule valeur ou 0 valeur Si 0 valeur la sous-requete vaut NULL

2.2. Exemple

On veut le (les) produit (s) le (les) plus cher (s)

```
SELECT id_produit, prix
FROM produit
WHERE prix = (
  SELECT MAX (prix)
  FROM produit
  )
```

2.3. Exemple 2:

les produits (id, description, prix) qui ont reçu au moins une satisfaction superieure à la moyenne des satisfactions

```
SELECT id_produit, des_produit, prix
FROM produit
NATURAL JOIN ligne_commande
WHERE satisfaction >= (
  SELECT
    AVG(satifaction
      FROM ligne_commande)
  )
```

3. Sous-requete corélée

3.1. exemple

Les lignes de commandes (nom_client, no°facture, id_produit) telles que la note de satisfaction est la plus mauvaise pour le produit

```
SELECT nom_client, satisfaction, id_produit
FROM ligne_commande
AS l
NATURAL JOIN facture
WHERE satisfaction = (
  SELECT
  MIN(satisfaction
    FROM ligne_commande
    WHERE id_produit = l.id_produit)
  )
```

On peut imaginer que pour chaque ligne de commande de ligne_commande NJ facture Postgre calcule la valeur de la sous-requere avec la valeur correspondante de id_produit $= sorte de boucle sur le calcule de la sous-requete

4. ALL, ANY

4.1. Syntaxe

Condition valeur op_comparaison ALL (SELECT ...) valeur op_comparaison ANY (SELECT ...)

op_comparaison : =, <, >, ... is DISTINCT FROM, ... Les sous-requete peut donner plusieurs valeurs

`ALL` : retourne true si à la comparaison de valeur avec chaque ligne du select retourne true Si 0 lignes dans le select => retourne true ANY: true si au moins une ligne comparée avec valeur donne true Si 0 ligne dans le select => retourne false

4.2. Exemple

les lignes de commandes qui ont la plus grande satisfction

```
SELECT *
FROM ligne_commande
WHERE satisfaction >=
  ALL(
    SELECT satisfaction
    FROM ligne_commande
    WHERE satisfaction IS NOT NULL)
```

4.3. Remarque

Si je ne mets pas la condition IS NOT NULL, des qu'une satisfaction est NULL, où aura 0 lignes de resultat pour la requete globale

Si je remplace >= ALL par > ANY

```
SELECT *
FROM ligne_commande
WHERE satisfaction >
  ANY(
    SELECT satisfaction
    FROM ligne_commande
    WHERE satisfaction IS NOT NULL)
```

ca rendra toutes les lignes où la satisfaction est au dessus de la plus mauvaise

5. IN et NOT IN

5.1. Syntaxe

valeur IN (SELECT ...) valeur NOT IN (SELECT ...) IN est equivalent à ANY NOT IN est equivalent à < > ALL (!= ALL)

5.2. Exemple

Les factures faites aux magasin ayant au moins 10 articles en stock.

Sous requête

les magasin ayant au moins 10 article en stock

```
SELECT id_magasin
FROM Stock
GROUP BY id_magesin
HAVING SUM(quantite) >= 10;
```

Requête final

```
SELECT *
FROM facture
WHERE id_magasin IN (
  SELECT id_magasin
  FROM Stock
  GROUP BY id_magesin
  HAVING SUM(quantite) >= 10
  );
```