# Notes & remarques de cours :

* Une création d’une BD n’est pas le rôle de l’admin normalement.
* Un schéma public est accessible par tus les utilisateurs de serveur de BD.
* ***Pgadmin*** n’est un client pour le serveur de base de données installé sur notre machine.
* ***Date*** est un type natif dans langage SQL et non pas composé comme on fait souvent.
* On peut définir de nouveaux types (ou domaines) ou plutôt alias avec possibilité de déterminer une contrainte sur la valeur.
* Plusieurs commandes SQL = on parle d’un script.
* Après création d’une DB par exemple, il faut rafraichir pour la voir
* PostgreSQL permet de créer une base sans clé primaire, mais y’a des pénalités (peut-être on peut pas ajouter de donnée ? )
* Unicité ou CHECK IN ne prend pas en compte NULL, vu qu’ils comparent des valeurs alors qu’avec NULL y’a même pas de valeur.
* En PostgreSQL, les dates entre double quottes stp.
* Une SGBD c’est plutôt un serveur de DB (Pgadmin n’est qu’un client pour PostgreSQL, même pour phpMyAdmin n’est qu’un outil d’administration de MySQL)
* Le SGBD réécrit les requêtes pour maximiser l’optimisation
* Le mot clé CASCADE permet de forcer certaines requêtes qu’on ne permet pas par défaut normalement, comme la suppression d’une table lié avec une autre)
* Utiliser serial instead of integer pour une clé primaire va mener la table à régénérer un ID unique à partir d’une séquence qui a été créé automatiquement.
* Dans les requêtes SQL, NULL n’est pas pris en compte sauf si mentionné.
* Lorsqu’on fait un SELECT, la recherche est contrôlé par un système d’indexage, les dernier tuples modifiés sont de dernier index à afficher (ce système suit celui des B-trees de recherche)
* Si on divise par exemple des petits nombre sur 1000, le résultat sera 0 vu qu’il a été tronqué, il suffit de diviser sur 1000.0 pour indiquer qu’on attend un réel comme réponse.
* Les calculs peuvent être effectuées soit sur la couche présentation (browser as example) ou application (server-side as example) ou même sur la couche données (SGBD), le choix dépend de différents paramètres selon les contraintes présentes.
* Syntaxe SQL :
  + ***SELECT*** nom\_colonne,valeur\_cte,appel à une fct, expression
  + ***[FROM]*** nom\_tble, nom\_vue, plusieurs tables
  + ***[WHERE]*** Prédicat AND/OR prédicat2
  + ***[ORDER BY]*** nom\_colonne
  + ***[LIMIT]*** nombre
  + ***[OFFSET]*** nombre
  + ***[GROUP BY] [HAVING]***
* Erreur de sémantique lors d’application des sous requêtes : on test seulement la compatibilité des types et pas du même attribut.
* Les jointures permettent de minimiser la taille de requêtes (par rapport au sous requêtes).
* Jointure dans condition est un produit cartésien.
* Il existe plusieurs types de jointures et la main différence est la performance qui est issue de la manière dont le SGBD performe ses actions (Nestedloops…etc. )
* Les VIEWS sert à créer une table exploitable sans qu’elle soit physiquement stockée sur le disque. (table virtuelle).
* The VIEW is not updateable. (ajout, suppression ..etc.)
* En PostgreSQL, il est possible de mettre à jour une base de données à travers une VIEW qui est issue d’elle.
* CREATE TABLE AS permet de créer des tables à partir des tables déjà existantes.
* Les règles gèrent l’ensemble d’effets des actions communes (DELETE,SELECT ..etc.) ce n’est pas une modification de source code, mais une sorte d’une couche externe that is overriding the old function.

***CREATE OR REPLACE FUNCTION GetSalleCapaciteSuperieurA(salle.capacite%Type)***

***RETURNS SETOF salle.batiment%Type***

***AS '***

***SELECT \* FROM Salle WHERE Capacite > $1;***

***'***

***LANGUAGE SQL;***

---------------------------------------------------------------------------------------------------------------------------------------------------------

***CREATE OR REPLACE FUNCTION DateReserBis (reservation.reservation\_id%TYPE)***

***RETURNS reservation.date\_resa%TYPE***

***AS***

***'***

***DECLARE***

***dateres reservation.date\_resa%TYPE;***

***BEGIN***

***SELECT INTO dateres date\_resa***

***FROM Reservation WHERE reservation\_id = $1;***

***/\* Le résultat de la requete est stockée dans dateres\*/***

***RETURN dateres;***

***END;***

***'***

***LANGUAGE 'plpgsql';***

---------------------------------------------------------------------------------------------------------------------------------------------------------

***CREATE OR REPLACE FUNCTION getNomPrenomFromLine(etudiant.etudiant\_id%TYPE)***

***RETURNS text***

***AS***

***'***

***DECLARE***

***monEtudiant etudiant%ROWTYPE;***

***NomPrenom text;***

***BEGIN***

***SELECT INTO monEtudiant \* FROM etudiant***

***WHERE etudiant\_id = $1;***

***NomPrenom := monEtudiant.nom || " " || monEtudiant.prenom;***

***RETURN NomPrenom;***

***END;***

***'***

***LANGUAGE 'plpgsql';***

***CREATE OR REPLACE FUNCTION getPhoneNumber(enseignant.enseignant\_id%TYPE)***

***RETURNS enseignant.telephone%TYPE***

***AS***

***'***

***DECLARE***

***phone enseignant.telephone%TYPE;***

***BEGIN***

***SELECT telephone INTO phone FROM enseignant***

***WHERE enseignant\_id = $1;***

***IF NOT FOUND THEN RAISE EXCEPTION "NOT FOUND";***

***ELSIF phone IS NULL THEN RAISE EXCEPTION "NULL";***

***ELSE RETURN phone;***

***END IF;***

***END;***

***'***

***LANGUAGE 'plpgsql';***

# Correction des exercices :

*Donner les numéros des fournisseurs qui approvisionnent l'usine n°1 en un produit rouge.*

SELECT nf

FROM puf

WHERE nu=1

AND np IN (SELECT np

FROM p

WHERE couleur='Rouge');

*Donner les numéros des produits qui sont livrés à toutes les usines de Casablanca*

* *Chercher les produits pour lesquels il n’existe pas d’usine de Casablanca pour laquelle il n’existe pas d’achat de ce produit.*

SELECT np

FROM p

WHERE NOT EXISTS

( SELECT \* FROM u WHERE ville=’CASABLANCA’ AND NOT EXISTS

( SELECT \* FROM puf WHERE puf.nu= u.nu AND p.np=puf.np));

*Donner les numéros des fournisseurs qui approvisione à la fois usine 1 et 2.*

SELECT nf

FROM puf

WHERE nu=1

AND nf IN (SELECT nf

FROM puf

WHERE nu=2);

*Donner les numéros des fournisseurs qui approvisionnent l’usine n°1 en un produit rouge. (En utilisant la jointure)*

SELECT nf

FROM puf,p

WHERE puf.np=p.np AND nu=1 AND couleur=’Rouge’

*Donner les numéros des fournisseurs qui approvisionnent une usine de Casablanca ou de Tanger en un produit rouge. (En utilisant la jointure)*

SELECT DISTINCT nomf

FROM puf,p,u,f

WHERE puf.np=p.np

AND puf.nu=u.nu

AND puf.nf=f.nf

AND (ville=’Casablanca’ OU ville=’Tanger’)

AND couleur=’Rouge’

*Donner les numéros des produits livrés à une usine par un fournisseur de la même ville*

SELECT np

FROM puf,u,f

WHERE puf.nf=f.nf

AND puf.nu=u.nu

AND nf.ville=u.ville

***Faire 9, 12,13 et 14! Et lire 7.2.1.1 (JOIN) de la documentation PostgreSQL***

*Donner les numéros des produits livrés à une usine par un fournisseur de la même ville*