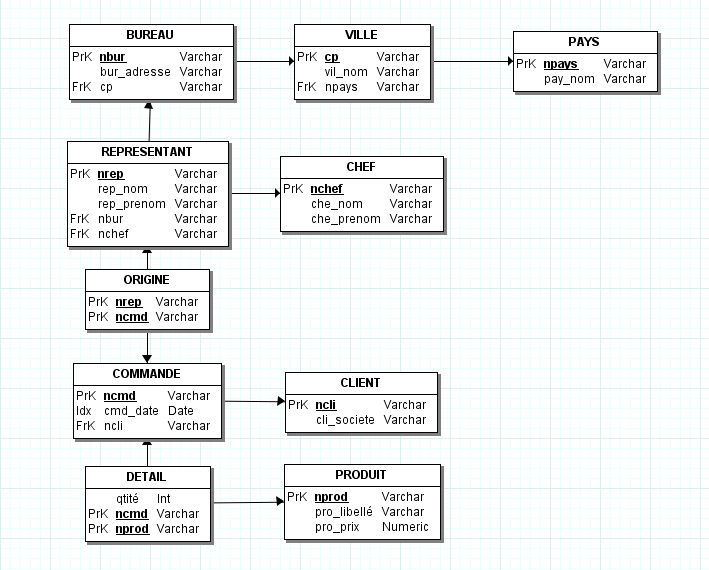


|  |
| --- |
| HIGH TECH COMPASS : 1 RUE  DU CENTRE 93160 NOISY LE GRAND  E-mail: administration@htcompass.fr – Web: [www.htcompass](http://www.htcompass).fr |

|  |
| --- |
| PL/pgSQL correction-02 |
| Présenté par : Prénom Nom |

**Exercices**

Note : Cette série d’exercices est accompagné du fichier : ‘*pgsql\_exo\_02\_données.sql*’, qui contient le scripte de création des tables ainsi que de leurs données.



**1. Faire une fonction qui renvoi le calcul de la tva (19.6) d’un champ renseigné en paramètre**

CREATE or replace FUNCTION calcul\_tva (a integer) RETURNS NUMERIC(5,2) AS $$

DECLARE

--déclaration de variable

BEGIN

--instruction

return a\*0.196;

END

$$ LANGUAGE 'plpgsql'

**2. Faire une procédure (sans paramètre) qui affiche les commandes du mois en cours**

--select \* from commande where date\_part('month',cmd\_date)='01';

Rq: date\_part('month',cmd\_date) retourne un double precision

CREATE or REPLACE function get\_mois\_enCours() returns setof record as $$

DECLARE

rec record;

BEGIN

for rec in

select cmd\_date

from commande

where date\_part('month',cmd\_date) = date\_part('month',current\_date)

loop

return next rec;

end loop;

return;

END;

$$ language 'plpgsql';

select cmd\_date from get\_mois\_enCours() as (cmd\_date date)

**3. Faire la même procédure qui affiche les commandes du mois en paramètre**

--select \* from commande where date\_part('month',cmd\_date)='01';

CREATE or replace function get\_mois(mm double precision) returns setof record as $$

DECLARE

rec record;

BEGIN

for rec in

select cmd\_date from commande where date\_part('month',cmd\_date) = mm

loop

return next rec;

end loop;

return;

End;

$$ language 'plpgsql';

select cmd\_date from get\_mois('02') as (cmd\_date date);

**4. Faire une procédure qui vérifie si une ville passé en paramètre existe dans la table VILLE (affichage dans la fenêtre des messages)**

CREATE FUNCTION rech\_ville (param1 text) RETURNS text AS $$

DECLARE

reponse varchar;

BEGIN

SELECT vil\_nom INTO reponse FROM ville WHERE vil\_nom = $1;

IF NOT FOUND THEN

reponse = reponse || ' not found';

raise notice '% not found',$1;

ELSE

reponse = reponse || ' found';

raise notice '% found',$1;

END IF;

return reponse;

END

$$ LANGUAGE 'plpgsql'

-- methode d’affichage

DO language plpgsql $$

BEGIN

execute rech\_ville('ATLANTA');

END

$$;

SELECT cmd\_date FROM rech\_ville('ATLANTA') as (vil\_nom text);

**5. Faire une procédure qui affiche les représentants qui réalise moins de deux commandes entre la date du jour et les 20 jours (ouvrés) passés.**

SELECT rep\_nom, count(cmd.ncmd)

FROM representant as rep

inner join origine as ori on rep.nrep=ori.nrep

inner join commande as cmd on ori.ncmd=cmd.ncmd

WHERE cmd\_date between current\_date and current\_date + integer '20'

and EXTRACT(DOW FROM cmd\_date) not in (6,7)

GROUP BY rep\_nom

**6. Faire une procédure qui vérifie et ajoute la colonne [jours] à la table COMMANDE .**

(à faire)

**7. Chercher sur internet la fonction qui détermine le jour de la semaine à une date passé en paramètre.**

Dow : Le jour de la semaine (de 0 à 6 ; dimanche étant le 0) uniquement pour les valeurs de type timestamp

SELECT EXTRACT (DOW FROM TIMESTAMP '2001-02-16 20:38:40');

Résultat : 5

Lundi=1, mardi=2,…, dimanche=0

Notez que la numérotation du jour de la semaine est différente de celui de la fonction to\_char.

**8. Utiliser cette fonction pour faire une procédure qui met à jour la colonne [jours] depuis la colonne cmd\_date de la table COMMANDE**