TD5 Fonctions en pl/pgsql

SQL Shell (psql)

```
postgres=# CREATE DATABASE hotels;
CREATE DATABASE
postgres=# \c hotels
Vous êtes maintenant connecté à la base de données « hotels » en tant qu'utilisateur « postgres ».
hotels=# \i 'C:/Users/lenny/Downloads/creer-base-hotels_local.sql'
```

Exercice 1 , *Requêtes SQL avec données nulles ou inexistantes*

Donnez les requêtes SQL permettant de trouver les informations suivantes.
Portez une attention particulière aux données nulles ou inexistantes.

1. Donner tous les logements qui n'apparaissent ni en tant qu'hôtel, ni en tant qu'appartement.

```
SELECT *
FROM logement l
WHERE l.numlogement NOT IN (SELECT h.numlogement FROM hotel h)
          AND l.numlogement NOT IN (SELECT  a.numlogement FROM appartement a);
```

```
numlogement |        nom         |    adresse     | ville | codepostal
------------+--------------------+----------------+-------+------------
          7 | B&B Quartier Latin | 10 rue d'Assas | Paris |      75005
(1 ligne)
```

2. Écrire une requête qui retourne, pour chaque logement, son nom, et un attribut `est_hotel` qui est `TRUE` si le logement donné en paramètre est un hôtel, `FALSE` si c'est un appartement, et `NULL` si ce n'est ni l'un ni l'autre.

```
SELECT l.nom,
       CASE
        WHEN ( l.numlogement IN (SELECT h.numlogement FROM hotel h) ) THEN 'TRUE'
        WHEN ( l.numlogement IN (SELECT a.numlogement FROM appartement a) ) THEN 'FALSE'
        WHEN (l.numlogement NOT IN (SELECT h.numlogement FROM hotel h)
                    AND l.numlogement NOT IN (SELECT  a.numlogement FROM appartement a))
                                                                        THEN 'NULL'
        END est_hotel
FROM logement l;
```

nom	est_hotel
Hotel des trois Lilas	TRUE
Hotel des Encyclopedistes	TRUE
Hotel des trois Philosophes	TRUE
Appartement des Ecoles	FALSE
Studio Erasme	FALSE
Appartement du Pantheon	FALSE
B&B Quartier Latin	NULL
Le Grand Hotel	TRUE

8 lignes)

3. Donner la note moyenne de chaque logement sur chaque critère.

Pour chaque hôtel, affichez les trois moyennes (une moyenne par critère) ainsi qu'une moyenne générale , avec un coefficient de 1/3 sur chacun des trois critères

(Que s'affiche-t-il si une des moyennes est NULL ?)

```
/* SQL: AVG with NULL Values :
 * stackoverflow.com/questions/22220449/sql-avg-with-null-values
 */
SELECT numlogement, nom, moyenne_proprete, moyenne_service, moyenne_situation,
      ( (moyenne_proprete + moyenne_service + moyenne_situation) / 3) AS
moyenne_generale
FROM (SELECT l.numlogement, l.nom,
       AVG(COALESCE(a.noteproprete,0)) AS moyenne_proprete,
       AVG(COALESCE(a.noteservice,0)) AS moyenne_service,
       AVG(COALESCE(a.notesituation,0)) AS moyenne_situation
      FROM logement l, hotel h
      LEFT JOIN avis a ON h.numlogement = a.numlogement
      WHERE l.numlogement = h.numlogement
      GROUP BY l.numlogement) AS moyennes;
```

```
numlogement |           nom            |    moyenne_proprete    |    moyenne_service     |    moyenne_situation   |    moyenne_generale
------------+--------------------------+------------------------+------------------------+------------------------+------------------------
          2 | Hotel des Encyclopedistes | 0.0000000000000000000 |     3.0000000000000000 |     3.0000000000000000 | 2.0000000000000000000
          3 | Hotel des trois Philosophes | 0.0000000000000000000 | 0.0000000000000000000 | 0.0000000000000000000 | 0.0000000000000000000
          1 | Hotel des trois Lilas     |     3.0000000000000000 |     2.6666666666666667 |     3.0000000000000000 |     2.8888888888888889
          8 | Le Grand Hotel            | 0.0000000000000000000 | 0.0000000000000000000 | 0.0000000000000000000 | 0.0000000000000000000
(4 lignes)
```

4. Afficher le nom des hôtels dont le nombre de réservations pour aujourd'hui dépasse le nombre de chambres.

```
SELECT l.nom, COALESCE(h.chambres,0) AS chambres, COUNT(r.numlogement) AS reservations
FROM logement l, hotel h, reserve r
WHERE r.datedebut = CURRENT_DATE AND l.numlogement = h.numlogement
      AND l.numlogement = r.numlogement
GROUP BY l.numlogement, h.numlogement
HAVING (COUNT(r.numlogement) > COALESCE(h.chambres,0));
```

Par exemple,

```
SELECT l.nom, COALESCE(h.chambres,0) AS chambres, COUNT(r.numlogement) AS reservations
FROM logement l, hotel h, reserve r
WHERE r.datedebut = '2020-02-25' AND l.numlogement = h.numlogement
      AND l.numlogement = r.numlogement
GROUP BY l.numlogement, h.numlogement
HAVING (COUNT(r.numlogement) > COALESCE(h.chambres,0));
```

```
           nom             | chambres | reservations
---------------------------+----------+--------------
 Hotel des Encyclopedistes |        5 |            6
(1 ligne)
```

Partie 2 , PL/pgSQL

Exercice 2 :

Écrire une fonction booléenne `est_hotel` qui retourne `TRUE` si le numéro de logement donné en paramètre est un hotel, `FALSE` si c'est un appartement, et `NULL` si c'est ni l'un ni l'autre. Si le logement n'existe pas, terminer avec une exception `RAISE 'Logement inexistant % ', num_logement USING ERRCODE='20002'`. (Voir Exercice 1 Question 2.) Vérifier les résultats avec `SELECT nom, est_hotel(numlogement) FROM logement;` Puis pour vérifier qu'un logement inexistant provoque bien une erreur `SELECT est_hotel(XXX);`, en remplaçant `XXX` par un numéro qui ne correspond pas à un logement.

```
CREATE OR REPLACE FUNCTION est_hotel(num_logement int)
RETURNS BOOLEAN AS $$
BEGIN
       CASE
        WHEN ( num_logement IN (SELECT h.numlogement FROM hotel h) ) THEN RETURN TRUE;
        WHEN ( num_logement IN (SELECT a.numlogement FROM appartement a) ) THEN RETURN FALSE;
        ELSE RAISE EXCEPTION 'Logement inexistant %', num_logement USING ERRCODE = '20002';
        END CASE;
END;
$$ LANGUAGE plpgsql;
```

```
hotels=# SELECT nom, est_hotel(numlogement) FROM logement;
ERREUR:  Logement inexistant 7
CONTEXTE : fonction PL/pgSQL est_hotel(integer), ligne 6 à RAISE
hotels=# SELECT nom, est_hotel(numlogement) FROM logement WHERE numlogement <> 7;
            nom             | est_hotel
----------------------------+-----------
 Hotel des trois Lilas      | t
 Hotel des Encyclopedistes  | t
 Hotel des trois Philosophes | t
 Appartement des Ecoles     | f
 Studio Erasme              | f
 Appartement du Pantheon    | f
 Le Grand Hotel             | t
(7 lignes)
```

Exercice 3 :

Écrire une fonction `reserver` qui prend en paramètre un numéro de logement `num_logement`, une date d'arrivée `date_debut` et de départ `date_fin` et crée une réservation pour `CURRENT_USER`. Si ce client n'existe pas, terminer avec une exception `RAISE 'Client inexistant % ', CURRENT_USER USING ERRCODE='20001'`. (Puis ajoutez-vous à la table `client`.) Si le logement n'existe pas, terminer avec une exception `RAISE 'Logement inexistant % ', num_logement USING ERRCODE='20002'`.

```
CREATE OR REPLACE FUNCTION reserver(num_logement int, date_debut date, date_fin date)
RETURNS SETOF Reserve AS $$
--  La fonction retourne un type “ensemble” (SETOF)
DECLARE
type_logement BOOLEAN;
ligne Client%ROWTYPE;
BEGIN
      -- Si le logement n'existe pas, termine avec une exception
      type_logement := est_hotel(num_logement);
      SELECT * INTO ligne FROM client WHERE login = CURRENT_USER;
      IF NOT FOUND THEN RAISE EXCEPTION 'Client inexistant % ', CURRENT_USER USING ERRCODE
= '20001' ; END IF;
      INSERT INTO reserve VALUES (CURRENT_USER, num_logement, date_debut, date_fin,
CURRENT_DATE);
       -- renvoie tous les reservations de CURRENT_USER
      RETURN QUERY SELECT * FROM reserve WHERE login = CURRENT_USER;
END;
$$ LANGUAGE plpgsql;
```

```
hotels=# SELECT reserver(888, '2020-11-11', '2020-12-12');
ERREUR:  Logement inexistant 888
CONTEXTE : fonction PL/pgSQL est_hotel(integer), ligne 6 à RAISE
fonction PL/pgSQL reserver(integer,date,date), ligne 8 à affectation
hotels=# SELECT reserver(8, '2020-11-11', '2020-12-12');
ERREUR:  Client inexistant postgres
CONTEXTE : fonction PL/pgSQL reserver(integer,date,date), ligne 10 à RAISE
hotels=# INSERT INTO client VALUES('postgres','pl','pgsql');
INSERT 0 1
```

```
hotels=# SELECT reserver(1, '2022-11-11', '2022-12-12');
                  reserver
---------------------------------------------
 (postgres,1,2022-11-11,2022-12-12,2022-02-19)
(1 ligne)


hotels=# SELECT reserver(8, '2023-11-11', '2023-12-12');
                  reserver
---------------------------------------------
 (postgres,1,2022-11-11,2022-12-12,2022-02-19)
 (postgres,8,2023-11-11,2023-12-12,2022-02-19)
(2 lignes)
```

Exercice 4 :

Écrire une fonction booléene `appartement_disponible` qui vérifie si un appartement est disponible aux dates demandées. Si le logement n'est pas un appartement, terminer avec une exception `RAISE 'Logement % n''est pas un appartement', num_logement USING ERRCODE='20101';` Indication : il suffit de vérifier qu'aucune réservation ne chevauche l'intervalle de dates demandé.

```
CREATE OR REPLACE FUNCTION appartement_disponible(num_logement int, date_debut date, date_fin date)
RETURNS BOOLEAN AS $$
DECLARE
is_not_appartement BOOLEAN;
ligne Reserve%ROWTYPE;
BEGIN
      is_not_appartement := est_hotel(num_logement);
      IF is_not_appartement THEN RAISE EXCEPTION 'Logement % nest pas un appartement', num_logement
USING ERRCODE = '20101'; END IF;
      SELECT * INTO ligne FROM reserve WHERE  numlogement = num_logement AND ( (date_debut >=
datedebut  AND date_debut <= datefin) OR (date_fin >= datedebut AND date_fin <= datefin) );
      IF NOT FOUND THEN RETURN true;
      ELSE RETURN false; END IF;
END;
$$ LANGUAGE plpgsql;
```

```
hotels=# INSERT INTO reserve VALUES ('diderot',4,'2023-11-11','2023-12-12', '2022-02-19');
INSERT 0 1
```

```
hotels=# SELECT * FROM reserve WHERE numlogement = 4;
  login   | numlogement | datedebut  |  datefin   | datereservation
---------+-------------+------------+------------+-----------------
 diderot |           4 | 2023-11-11 | 2023-12-12 | 2022-02-19
(1 ligne)
```

```
hotels=# SELECT appartement_disponible(1,'2023-12-13', '2023-12-25');
ERREUR:  Logement 1 nest pas un appartement
CONTEXTE : fonction PL/pgSQL appartement_disponible(integer,date,date), ligne 7 à RAISE
hotels=# SELECT appartement_disponible(11,'2023-12-13', '2023-12-25');
ERREUR:  Logement inexistant 11
CONTEXTE : fonction PL/pgSQL est_hotel(integer), ligne 6 à RAISE
fonction PL/pgSQL appartement_disponible(integer,date,date), ligne 6 à affectation
```

```
hotels=# SELECT appartement_disponible(4,'2023-12-11', '2023-12-25');
 appartement_disponible
------------------------
 f
(1 ligne)


hotels=# SELECT appartement_disponible(4,'2023-12-12', '2023-12-25');
 appartement_disponible
------------------------
 f
(1 ligne)


hotels=# SELECT appartement_disponible(4,'2023-12-13', '2023-12-25');
 appartement_disponible
------------------------
 t
(1 ligne)
```

```
hotels=#  SELECT appartement_disponible(4,'2023-10-11', '2023-11-11');
 appartement_disponible
------------------------
 f
(1 ligne)


hotels=# SELECT appartement_disponible(4,'2023-10-11', '2023-11-10');
 appartement_disponible
------------------------
 t
(1 ligne)


hotels=# SELECT appartement_disponible(4,'2023-10-11', '2023-11-25');
 appartement_disponible
------------------------
 f
(1 ligne)


hotels=# SELECT appartement_disponible(4,'2023-12-11', '2023-12-25');
 appartement_disponible
------------------------
 f
(1 ligne)
```

Exercice 5 :

Écrire une fonction booléene `hotel_disponible` qui vérifie si un hôtel est disponible à une date donnée. Si le logement n'est pas un hotel, terminer avec une exception `RAISE 'Logement % n''est pas un hotel', num_logement USING ERRCODE='20102';`

```
CREATE OR REPLACE FUNCTION hotel_disponible(num_logement int, date_hotel date)
RETURNS BOOLEAN AS  $$
DECLARE
is_hotel BOOLEAN;
nb_chambres int;
result RECORD; -- RECORD : n-uplet de structure arbitraire
BEGIN
      is_hotel := est_hotel(num_logement);
      IF NOT is_hotel THEN RAISE EXCEPTION 'Logement % nest pas un hotel', num_logement USING ERRCODE =
'20102'; END IF;

      SELECT COALESCE(chambres,0) INTO nb_chambres FROM hotel WHERE numlogement = num_logement;
      IF nb_chambres < 1 THEN RETURN false; END IF;

      SELECT COALESCE(h.chambres,0) AS chambres, COUNT(r.numlogement) AS reservations INTO result
      FROM hotel h
      LEFT JOIN reserve r ON h.numlogement = r.numlogement
      WHERE date_hotel >= datedebut  AND date_hotel <= datefin
            AND h.numlogement = num_logement
      GROUP BY h.numlogement
      HAVING (COUNT(r.numlogement) >= COALESCE(h.chambres,0));

      IF NOT FOUND THEN RETURN true;
      ELSE RETURN false; END IF;
END;
$$ LANGUAGE plpgsql;
```

```
hotels=# SELECT l.nom, COALESCE(h.chambres,0) AS chambres, COUNT(r.numlogement) AS reservations
hotels-# FROM logement l, hotel h, reserve r
hotels-# WHERE r.datedebut = '2020-02-25' AND l.numlogement = h.numlogement
hotels-#       AND l.numlogement = r.numlogement
hotels-# GROUP BY l.numlogement, h.numlogement
hotels-# HAVING (COUNT(r.numlogement) > COALESCE(h.chambres,0));
           nom            | chambres | reservations
--------------------------+----------+--------------
 Hotel des Encyclopedistes |        5 |            6
(1 ligne)


hotels=# SELECT hotel_disponible(2, '2020-02-25');
 hotel_disponible
------------------
 f
(1 ligne)


hotels=# SELECT hotel_disponible(2, '2020-03-17');
 hotel_disponible
------------------
 t
(1 ligne)


hotels=# SELECT hotel_disponible(3, '2025-10-03');
 hotel_disponible
------------------
 f
(1 ligne)
```

Exercice 6 :

Écrire une fonction `disponible` qui vérifie si un logement est disponible aux dates demandées. Pour les appartements, appeler la fonction `appartement_disponible` sur l'intervalle, et pour les hotels, appeler la fonction `hotel_disponible` sur chaque jour de l'intervalle.

```
CREATE OR REPLACE FUNCTION disponible(num_logement int, date_debut date, date_fin date)
RETURNS BOOLEAN AS $$
DECLARE
is_hotel BOOLEAN;
date_i date;
check_result BOOLEAN;
BEGIN
      is_hotel := est_hotel(num_logement);
      -- appartement_disponible(num_logement int, date_debut date, date_fin date)
      IF NOT is_hotel THEN RETURN appartement_disponible(num_logement, date_debut,
date_fin);
      ELSE
            date_i := date_debut;
            check_result := true;
            WHILE (date_i <= date_fin AND check_result = true)
            LOOP
                  -- hotel_disponible(num_logement int, date_hotel date)
                  IF NOT hotel_disponible(num_logement, date_i) THEN
                        check_result := false;
                  END IF;
                  date_i := (date_i + 1);
            END LOOP;
      END IF;

      RETURN check_result;
END;
$$ LANGUAGE plpgsql;
```

Exercice 7 :

Reprendre la fonction `reserver` et intégrer la validation des dates. Si les dates sont invalides, terminer avec une exception `RAISE 'Reservation impossible pour les dates % % ', date_debut, date_fin USING ERRCODE='20003'`. Si tout se passe correctement, terminer avec `RAISE NOTICE 'Reservation faite.'`

```
CREATE OR REPLACE FUNCTION reserver_bis(num_logement int, date_debut date, date_fin date)
RETURNS void AS $$
DECLARE
type_logement BOOLEAN;
ligne Client%ROWTYPE;
BEGIN
      -- Si le logement n'existe pas, termine avec une exception
      type_logement := est_hotel(num_logement);

      IF NOT ( disponible(num_logement, date_debut, date_fin) ) THEN
            RAISE 'Reservation impossible pour les dates % % ', date_debut, date_fin USING
ERRCODE='20003';
      END IF;

      SELECT * INTO ligne FROM client WHERE login = CURRENT_USER;
      IF NOT FOUND THEN RAISE EXCEPTION 'Client inexistant % ', CURRENT_USER USING ERRCODE = '20001' ;
END IF;
      INSERT INTO reserve VALUES (CURRENT_USER, num_logement, date_debut, date_fin, CURRENT_DATE);

      RAISE NOTICE 'Reservation faite.';
END;
$$ LANGUAGE plpgsql;
```

Exercice 8 :

Écrire une fonction booléenne `reservation_sure` qui prend en paramètre le nom et prénom de client, date d'arrivée, date de départ, le numéro de logement.
Si la table `client` ne contient pas de client avec le même nom et prénom alors la fonction ajoutera le nouveau client dans la table `client`, comme son `login` on prendra le mot obtenu en concaténant son nom et prénom. (Rappel : `a||b` donne la concaténation de `a` et `b`.) Si la table `client` contient plusieurs clients ayant le même nom et prénom alors on lance une exception.
Si la table client contient exactement un client avec le même nom et prénom alors on suppose que c'est le client qui fait la réservation.
La réservation est impossible s'il existe un jour dans la période donnée tel que

- il n'y a aucune chambre libre si le logement est un hôtel ou
- il existe un jour tel que l'appartement est réservé si le logement est un appartement.

Si la réservation est impossible lancer une exception.
Sinon faites la réservation.