**TP2**

**A - Les requêtes SQL**

Créer la foreign table de personne\_insee :

CREATE FOREIGN TABLE personne\_insee (nomprenom character varying(100), sexe character(1), datenaiss character varying(10), lieunaiss character varying(10), commnaiss character varying(80), paysnaiss character varying(50), datedeces character varying(10), lieudeces character varying(10), actedeces character varying(50)) SERVER remote\_insee\_deces;

CREATE TABLE personne\_insee\_small AS SELECT \* FROM personne\_insee;

CREATE TABLE personne\_insee\_medium AS SELECT \* FROM personne\_insee;

CREATE TABLE personne\_insee\_large AS SELECT \* FROM personne\_insee;

A1 - Focus sur les anomalies

Question 1 : Sans exécuter Qa, expliquer en français de que retourne la requête Qa.

La requête Qa retourne les 10 combinaisons uniques de nomprenom, datenaiss, lieunaiss, datedeces, lieudeces qui ont le plus grand nombre d'occurrences dans la table personne\_insee.

Question 2 : Définir Qb à partir de Qa pour obtenir le plan d'exécution de celle-ci. Exécuter Q0b, puis analyser le résultat retourné.

explain (analyse, verbose, costs, buffers, timing, summary) SELECT nomprenom, datenaiss, lieunaiss, datedeces, lieudeces, count(\*) as nb FROM personne\_insee GROUP BY nomprenom, datenaiss, lieunaiss, datedeces, lieudeces ORDER BY nb DESC LIMIT 10;

Question 3 : Définir le prédicat *is\_date* qui prend en argument une chaîne de caractères et qui retourne *true* si l'argument correspond à une date valide, et false sinon.

CREATE OR REPLACE FUNCTION is\_date(date\_string TEXT) RETURNS BOOLEAN AS $$

DECLARE

year\_part INTEGER;

month\_part INTEGER;

day\_part INTEGER;

BEGIN

-- Vérifie si la chaîne de date est non nulle

IF date\_string IS NULL OR date\_string = '' OR LENGTH(date\_string) != 8 THEN

RETURN false;

END IF;

-- Extrait les parties de l'année, du mois et du jour de la chaîne de date

year\_part := CAST(SUBSTRING(date\_string FROM 1 FOR 4) AS INTEGER);

month\_part := CAST(SUBSTRING(date\_string FROM 5 FOR 2) AS INTEGER);

day\_part := CAST(SUBSTRING(date\_string FROM 7 FOR 2) AS INTEGER);

-- Vérifie que l'année, le mois et le jour sont valides

IF year\_part < 1 OR year\_part > 9999 THEN

RETURN FALSE;

END IF;

IF month\_part < 1 OR month\_part > 12 THEN

RETURN FALSE;

END IF;

IF day\_part < 1 OR day\_part > 31 THEN

RETURN FALSE;

END IF;

-- Essaye de convertir la chaîne en une date au format 'YYYYMMDD'

BEGIN

-- Utilisez la fonction to\_date pour essayer de convertir la chaîne en date

PERFORM to\_date(date\_string, 'YYYYMMDD');

-- Si la conversion réussit, retournez true

RETURN TRUE;

EXCEPTION

WHEN others THEN

-- En cas d'erreur de conversion, retournez false

RETURN FALSE;

END;

END;

$$ LANGUAGE plpgsql;

Question 4 : Définir la requête Qc qui retourne le triplet *(nbT, nbDNE, ratio)* où :

* *nbT* correspond au nombre total de tuples de la table *personne\_insee*,
* *nbDNE* correspond au nombre de tuples avec une Date de Naissance Erronée
* *ratio* correspond à la proportion de tuples ayant une date de naissance erronée.

SELECT

COUNT(\*) AS nbT,

SUM(CASE WHEN is\_date(datenaiss) = false THEN 1 ELSE 0 END) AS nbDNE,

(SUM(CASE WHEN is\_date(datenaiss) = false THEN 1 ELSE 0 END)::float / COUNT(\*)) AS ratio

FROM personne\_insee\_medium;

nbt | nbdne | ratio

----------+-------+-----------------------

24916669 | 89199 | 0.0035798926413478462

Question 5 : Définir la requête Qd qui retourne le nombre de personnes sans nom de famille

SELECT COUNT(\*) AS nombre\_personnes\_sans\_nom

FROM personne\_insee\_small

WHERE split\_part(nomprenom, '\*', 1) = '';

Réponse : 95

A2 - Autres requêtes SQL

Q1 **Pour chaque département de décès, donner le nombre de personnes ayant au moins un doublon de déclaration de décès**. Le résultat sera trié par ordre décroissant du nombre de doublon. Pour cela, vous pouvez vous inspirer de la requête Qa, en sachant que :

* vous aurez à ne garder que les personnes apparaissant plus d'une fois;
* extraire le code département du code INSEE de la commune de décès. Attention aux codes de département sur 3 chiffres des territoires d'outre-mer! (NB: un 'case when' pourrait vous être utile).

SELECT

SUBSTRING(lieudeces FROM 1 FOR 2) AS code\_departement,

COUNT(\*) AS nombre\_personnes\_avec\_doublon

FROM personne\_insee\_medium

WHERE (nomprenom, datenaiss, lieunaiss) IN (

SELECT nomprenom, datenaiss, lieunaiss

FROM personne\_insee\_medium

WHERE lieudeces IS NOT NULL -- Filter out records without a place of death

GROUP BY nomprenom, datenaiss, lieunaiss

HAVING COUNT(\*) > 1

)

GROUP BY code\_departement

ORDER BY nombre\_personnes\_avec\_doublon DESC;

—----------------------------------------------------------------

SELECT CASE WHEN lieudeces LIKE '97%' THEN SUBSTRING(lieudeces, 0, 4) ELSE SUBSTRING(lieudeces, 0, 3) END as numdep, COUNT(\*) AS nbD FROM

(

SELECT nomprenom, datenaiss, lieunaiss, datedeces, lieudeces, COUNT(\*) as nbD

FROM personne\_insee\_medium P

GROUP BY nomprenom, datenaiss, lieunaiss, datedeces, lieudeces

HAVING COUNT(\*) > 1

) t

GROUP BY numdep

ORDER BY COUNT(\*) DESC;

**Q2. Pour chaque décennie, donnez le top 10 des noms de famille les plus portés**

WITH SS1 AS (

SELECT

generate\_series('1970-01-01'::date, CURRENT\_DATE, '1 decade'::interval) AS decades\_series

),

SS2 AS (

SELECT

split\_part(nomprenom, '\*', 1) AS nom,

EXTRACT(YEAR FROM TO\_DATE(datenaiss, 'YYYYMMDD'))::integer AS annee\_naissance,

EXTRACT(YEAR FROM TO\_DATE(datedeces, 'YYYYMMDD'))::integer AS annee\_deces

FROM

personne\_insee\_medium

WHERE

is\_date\_immutable(datenaiss) AND is\_date\_immutable(datedeces)

),

RankedNames AS (

SELECT

EXTRACT(YEAR FROM SS1.decades\_series) AS decades,

SS2.nom,

COUNT(\*) AS occurrences,

ROW\_NUMBER() OVER (PARTITION BY EXTRACT(YEAR FROM SS1.decades\_series) ORDER BY COUNT(\*) DESC) AS rank

FROM SS1

JOIN SS2 ON

EXTRACT(YEAR FROM SS1.decades\_series) BETWEEN SS2.annee\_naissance AND SS2.annee\_deces

GROUP BY SS1.decades\_series, SS2.nom

)

SELECT

decades,

STRING\_AGG(rank || '. ' || nom || ' (' || occurrences || ')', ' ; ') AS classement

FROM RankedNames

WHERE rank <= 10

GROUP BY decades

ORDER BY decades;

**Q3. Quel est la durée de vie moyenne des personnes selon leur département de naissance ?**

WITH Personnes AS (

SELECT

nomprenom,

CASE WHEN SUBSTRING(lieunaiss FROM 1 FOR 2) = '97' THEN SUBSTRING(lieunaiss FROM 1 FOR 3) ELSE SUBSTRING(lieunaiss FROM 1 FOR 2) END AS dep\_naissance,

CASE WHEN SUBSTRING(lieudeces FROM 1 FOR 2) = '97' THEN SUBSTRING(lieudeces FROM 1 FOR 3) ELSE SUBSTRING(lieudeces FROM 1 FOR 2) END AS dep\_deces,

(TO\_DATE(datedeces, 'YYYYMMDD') - TO\_DATE(datenaiss, 'YYYYMMDD')) AS duree\_vie

FROM personne\_insee\_medium

WHERE is\_date\_immutable(datenaiss) AND is\_date\_immutable(datedeces)

)

SELECT d.dep, d.nom, COUNT(\*) AS nombre\_personnes\_deces, justify\_interval((AVG(duree\_vie) || ' days')::interval ) AS esperance\_vie\_moyenne

FROM Personnes p JOIN remote\_departement d ON p.dep\_naissance = d.dep

GROUP BY d.dep, d.nom

ORDER BY esperance\_vie\_moyenne DESC, d.dep;

**B - Traitement des requêtes**

**INDEX Q1**

CREATE INDEX index\_q1 ON personne\_insee(nomprenom);

**INDEX Q2**

CREATE OR REPLACE FUNCTION is\_date\_immutable(date\_string TEXT) RETURNS BOOLEAN AS $$

DECLARE

year\_part INTEGER;

month\_part INTEGER;

day\_part INTEGER;

BEGIN

-- Vérifie si la chaîne de date est non nulle

IF date\_string IS NULL OR date\_string = '' OR LENGTH(date\_string) != 8 THEN

RETURN false;

END IF;

-- Extrait les parties de l'année, du mois et du jour de la chaîne de date

year\_part := CAST(SUBSTRING(date\_string FROM 1 FOR 4) AS INTEGER);

month\_part := CAST(SUBSTRING(date\_string FROM 5 FOR 2) AS INTEGER);

day\_part := CAST(SUBSTRING(date\_string FROM 7 FOR 2) AS INTEGER);

-- Vérifie que l'année, le mois et le jour sont valides

IF year\_part < 1 OR year\_part > 9999 THEN

RETURN FALSE;

END IF;

IF month\_part < 1 OR month\_part > 12 THEN

RETURN FALSE;

END IF;

IF day\_part < 1 OR day\_part > 31 THEN

RETURN FALSE;

END IF;

-- Essaye de convertir la chaîne en une date au format 'YYYYMMDD'

BEGIN

-- Utilisez la fonction to\_date pour essayer de convertir la chaîne en date

PERFORM to\_date(date\_string, 'YYYYMMDD');

-- Si la conversion réussit, retournez true

RETURN TRUE;

EXCEPTION

WHEN others THEN

-- En cas d'erreur de conversion, retournez false

RETURN FALSE;

END;

END;

$$ LANGUAGE plpgsql IMMUTABLE;

**CREATE INDEX index\_q2 ON personne\_insee\_medium(datenaiss, datedeces) WHERE is\_date\_immutable(datenaiss) AND is\_date\_immutable(datedeces);**

**INDEX Q3**

CREATE INDEX index\_q3 ON personne\_insee (nomprenom, datenaiss, datedeces) WHERE is\_date\_immutable(datenaiss) AND is\_date\_immutable(datedeces);

shared\_buffers : 1GB au lieu de 128MB

effective\_cache\_size : 4GB