-------------------- PROJET TRANSVERSE - MBA BIG DATA - SQL --------------------

-- GROUPE : Mbery, Chloé, Nasr Eddine, Nour Eddine et Yousra

-- [1] ETUDE GLOBALE

-- (A) Répartition client selon 5 critères par ordre de priorité

-- 1) VIP : le client est VIP si vip=1

-- 2) NEW\_N2 : le client a adhéré au cours de l'année N-2

-- 3) NEW\_N1 : le client a adhéré au cours de l'année N-1

-- 4) ADHÉRENT : le client est toujours en cours d'adhésion (date de fin d'adhésion > 2018/01/01)

-- 5) CHURNER : le client a churné (date de fin d'adhésion < 2018/01/01)

-- Note : le critère le plus au-dessus est prioritaire

-- exemple : un client étant VIP, et ayant adhéré sur l'année N-1 sera compté comme étant VIP

-- Pour cela, on va créer une colonne intermédiaire 'mix\_criteres'

alter table client add mix\_criteres varchar(50);

-- Dans laquelle on pourra stocker chaque critère au fur et à mesure que la condition associée est validée

-- On commence par remplir toutes les lignes qui valident la condition VIP

-- Puis on complète les lignes vides par la condition NEW\_N2, et ainsi de suite jusqu'à la condition CHURNER

-- Et pour respecter l'ordre des prio, dès qu'un critère est validé, on ne peux plus l'écraser

update client set mix\_criteres = (case

when vip = 1 then 'VIP'

when vip != 1 and extract (year from datedebutadhesion) = 2017 then 'NEW\_N1'

when vip != 1 and extract (year from datedebutadhesion) = 2016 then 'NEW\_N2'

when vip != 1 and extract (year from datedebutadhesion) != 2016 and extract (year from datedebutadhesion) != 2017 and datefinadhesion > '2018-01-01' then 'ADHÉRENT'

when vip != 1 and extract (year from datedebutadhesion) != 2016 and extract (year from datedebutadhesion) != 2017 and datefinadhesion < '2018-01-01' then 'CHURNER'

else null end);

-- Vérification overall et critère par critère

select \* from client;

select distinct VIP, extract (year from datedebutadhesion) as debut, extract (year from datefinadhesion) as fin, mix\_criteres from client

where mix\_criteres = 'VIP';

select distinct VIP, extract (year from datedebutadhesion) as debut, extract (year from datefinadhesion) as fin, mix\_criteres from client

where mix\_criteres = 'NEW\_N1';

select distinct VIP, extract (year from datedebutadhesion) as debut, extract (year from datefinadhesion) as fin, mix\_criteres from client

where mix\_criteres = 'NEW\_N2';

select distinct VIP, extract (year from datedebutadhesion) as debut, extract (year from datefinadhesion) as fin, mix\_criteres from client

where mix\_criteres = 'ADHÉRENT';

select distinct VIP, extract (year from datedebutadhesion) as debut, extract (year from datefinadhesion) as fin, mix\_criteres from client

where mix\_criteres = 'CHURNER';

-- (B) Comportement du CA global par client N-2 versus N-1

-- On identifie les données de la table où se trouve la notion de chiffre d'affaires:

select \* from entete\_ticket limit 10;

-- La table contient des données de 2016 et 2017 uniquement

select distinct extract(year from tic\_date) from entete\_ticket;

-- Somme du CA par client par année

select sum(tic\_totalttc) as CA\_global, idclient, extract(year from tic\_date) as annee from entete\_ticket

group by idclient, annee order by idclient, annee;

-- Jointure avec la table client, CA global par client et par année

select sum(tic\_totalttc) as CA\_global, client.idclient, extract(year from tic\_date) as annee, mix\_criteres from entete\_ticket

join client on client.idclient = entete\_ticket.idclient

where extract(year from tic\_date) between 2016 and 2017

group by client.idclient, annee order by client.idclient, annee;

-- Cela peut etre intéressant aussi de voir l'évolution du CA d'une année sur l'autre par groupe de clients (VIP, CHURNER...)

-- Jointure avec la table client, CA global par critère et par année

select sum(tic\_totalttc) as CA\_global, mix\_criteres, extract(year from tic\_date) as annee from entete\_ticket

join client on client.idclient = entete\_ticket.idclient

where extract(year from tic\_date) between 2016 and 2017

group by mix\_criteres, annee order by mix\_criteres, annee;

-- (C) Répartition client par age et par sexe

-- On vient créer deux nouvelles colonnes: age et sexe

-- La colonne age permet de calculer l'age d'un client en fonction de sa date de naissance et de la date du jour

-- Et la colonne sexe permet d'attribuer un sexe à une civilité

-- Création de la colonne sexe

alter table client add sexe varchar(50);

-- Cette colonne permet d'attribuer un genre ou un sexe à une civilité

-- Pour cela, on doit savoir combien de civilités différentes sont utilisées dans la base de données ainsi que leur nom

select count(distinct civilite) from client;

select distinct civilite from client;

-- Il y a donc 6 civilités qui sont clairement identifiables: 3 pour les femmes et 3 pour les hommes

update client set sexe = (case

when civilite = 'monsieur' then 'homme'

when civilite = 'MONSIEUR' then 'homme'

when civilite = 'Mr' then 'homme'

when civilite = 'madame' then 'femme'

when civilite = 'MADAME' then 'femme'

when civilite = 'Mme' then 'femme'

else null end);

-- Vérification

select distinct civilite, sexe from client;

-- Création de la colonne age

alter table client add age integer;

update client set age = extract (year from age(now(),datenaissance));

select \* from client;

select distinct age from client order by age;

-- NETTOYAGE DES DONNEES

-- Il y a 845 876 clients, tous avec des id différents

select count(\*) from client;

select count(distinct idclient) from client;

-- age theorique, age reel

-- CAS ERREUR: AGE NULL

-- Certains clients n'ont pas renseigné leur date de naissance

-- On se retrouve alors avec des erreurs dans le calcul de l'age

-- 507 902 clients ont une date de naissance, soit 60%

select count(\*) from client where age is not null;

-- Est-il prudent de supprimer 40% de la base de données sans en connaitre l'impact sur le reste des analyses?

-- Si on prend l'exemple de l'analyse par critere faite précedemment, on a:

-- 35% de CHURNERs si on prend toute la base versus 32% si on prend uniquement les ages valides

select count(\*) from client where mix\_criteres = 'CHURNER';

select count(\*) from client where mix\_criteres = 'CHURNER' and age is not null;

-- 13% de VIP si on prend toute la base, et 13% aussi si on prend uniquement les ages valides

select count(\*) from client where mix\_criteres = 'VIP';

select count(\*) from client where mix\_criteres = 'VIP' and age is not null;

-- 14% de VIP si on prend toute la base versus 18% aussi si on prend uniquement les ages valides

select count(\*) from client where mix\_criteres = 'NEW\_N2';

select count(\*) from client where mix\_criteres = 'NEW\_N2' and age is not null;

-- On choisira de ne pas supprimer les clients qui n'ont pas rempli leur date de naissance pour ne pas perdre en précision dans l'analyse par critère

-- PENDING... Une idée serait de créer une colonne 'complete\_data' pour indiquer les clients qui ont des données complètes

-- CAS ABSURDE NUMERO 1 : AGE NEGATIF OU NUL

-- On retrouve parfois des valeurs négatives, ce qui n'a pas vraiment de sens

-- C'est le cas pour les dates de naissance qui ont une date postérieure à la date du jour

-- Plusieurs solutions: supprimer ces clients, laisser un age négatif, remplacer les valeurs négatives...

-- Avant de choisir une solution, il faut savoir combien de clients sont concernés pour connaitre l'impact sur l'ensemble des données

-- Si peu de clients sont concernés, on pourra les supprimer

-- Mais si le nombre de clients est trop significatif, alors il faudra rajouter une valeur null

-- Ainsi, on les exclura de la répartition par age et par sexe, mais on conservera leurs données pour d'autres analyses

-- Et parmi eux, 63 sont des clients avec un age négatif ou nul

select count(\*) from client where age < 0 or age = 0;

-- Il semble évident que si nous supprimons ces 60 clients de la base de données, cela n'aura que peu voire aucun impact

-- CAS RGPD : PROTECTION DES DONNEES PERSONNELLES DES MINEURS

-- Maintenant, d'un point de vue business, quelle que soit la société, cela parait absurde d'avoir des clients agés d'1 ou 6 ans

-- C'est raisonnable se dire qu'un client devrait etre majeur pour etre encarté et supprimer alors tous les clients mineurs

-- Or d'un point de vue RGPD en France, il existe 2 cas de mineurs selon l'article 8 du RGPD

-- A partir de 16 ans, ils peuvent consentir librement au traitement de données personnelles

-- Les moins de 16 ans pour qui le consentement doit etre donné par le mineur et son représentant légal

-- C'est un double consentement qui doit etre libre, spécifique, éclairé et univoque

-- Combien de clients ont moins de 16 ans? 621 (incl. 63 cas absurdes)

select count(\*) from client where age < 16;

-- On admettra qu'il faut avoir plus de 16 ans dans notre société pour souscrire une carte de fidélité

-- On choisit donc de supprimer les 780 clients dont l'age est strictement inférieur à 16 ans

delete from client where age < 16;

-- Vérification: 0 clients dont l'age est inférieur à 16 ans, il reste 845 096 clients dans la base de données

select count(\*) from client where age < 16;

select count(\*) from client;

-- CAS ABSURDE NUMERO 2 : AGE TROP GRAND

-- Le dernier cas à étudier est celui où l'age est beaucoup trop grand

-- On retrouve des valeurs allant de 100 à 250 ans, 360 clients sont concernés

select count(age) from client where age > 100;

-- Mais quel seuil d'age maximal fixer?

-- Le record français est de 122 ans

-- On peut donc supposer que tous les 40 clients au dessus de 122 ans sont des erreurs et les supprimer

select count(age) from client where age > 122;

delete from client where age > 122;

-- Tableau de répartition du nombre de clients par age et par sexe

select count(\*) as nb\_clients, sexe, age from client group by sexe, age order by age DESC, sexe;

-- [2] ETUDE PAR MAGASIN

-- (A) Résultat par magasin, et une ligne pour le total

-- Création de la table Magasin

create table magasin ();

-- Code magasin

alter table magasin add code\_mag varchar(3);

insert into magasin (code\_mag) select codesociete from ref\_magasin ;

-- nb de clients rattachés par magasin (avec une color\_bar en fonction de la quantité)

alter table magasin add nb\_client int;

update magasin set nb\_client = (select count(idclient) from client

where magasin.code\_mag = client.magasin group by magasin);

-- nb de clients actifs sur N-2

alter table magasin add nb\_client\_2016 int;

update magasin set nb\_client\_2016 = (select count(distinct(idclient)) from entete\_ticket

where magasin.code\_mag = entete\_ticket.mag\_code and extract(year from tic\_date) = 2016);

-- nb de clients actifs sur N-1

alter table magasin add nb\_client\_2017 int;

update magasin set nb\_client\_2017 = (select count(distinct(idclient)) from entete\_ticket

where magasin.code\_mag = entete\_ticket.mag\_code and extract(year from tic\_date) = 2017);

-- % de clients N-2 vs N-1 (en vert si positif et en rouge si negatif)

alter table magasin add evol\_nb\_client decimal(13,2);

update magasin

set evol\_nb\_client = ((nb\_client\_2017 - nb\_client\_2016)/(NULLIF(nb\_client\_2016,0) ::FLOAT))\*100 ::FLOAT;

-- total\_ttc N-2

alter table magasin add CA\_ttc\_2016 decimal(13,2);

update magasin set CA\_ttc\_2016 = (select sum(tic\_totalttc) from entete\_ticket

where magasin.code\_mag = entete\_ticket.mag\_code and extract(year from tic\_date) = 2016);

-- total\_ttc N-1

alter table magasin add ca\_ttc\_2017 decimal(13,2);

update magasin set ca\_ttc\_2017 = (select sum(tic\_totalttc) from entete\_ticket

where magasin.code\_mag = entete\_ticket.mag\_code and extract(year from tic\_date) = 2017);

-- différence entre N-2 et N-1 (en vert si positif et en rouge si negatif)

alter table magasin add diff\_ca\_ttc decimal(13,2);

update magasin set diff\_ca\_ttc = ca\_ttc\_2017 - ca\_ttc\_2016 ::FLOAT;

-- % de clients N-2 vs N-1 (en vert si positif et en rouge si negatif)

alter table magasin add evol\_ca\_ttc decimal(13,2);

update magasin set evol\_CA\_ttc = (diff\_CA\_ttc/(NULLIF(ca\_ttc\_2016,0) ::FLOAT))\*100 ::FLOAT;

-- icône de satisfaction : positif si %client actif évolue et total TTC aussi

-- négatif si diminution des 2 indicateurs, moyen seulement l'un des deux diminue

-- pour info, il n'y a pas de cas d'évolution nulle

alter table magasin add satisfaction varchar(50);

update magasin set satisfaction = (case

when evol\_nb\_client > 0 and evol\_CA\_ttc > 0 then '1 POSITIVE'

when evol\_nb\_client > 0 and evol\_CA\_ttc < 0 or evol\_nb\_client < 0 and evol\_CA\_ttc > 0 then '0 MOYENNE'

when evol\_nb\_client < 0 and evol\_CA\_ttc < 0 then '-1 NEGATIVE'

else null end);

-- afficher la table magasin triée par ordre de satisfaction

select \* from magasin order by satisfaction DESC;

-- CAS du magasin EST

-- ce code magasin apparait dans la table client mais pas dans la table entete\_ticket

select \* from magasin where code\_mag = 'EST';

select \* from client where magasin = 'EST';

select count(\*) from client where magasin = 'EST';

select distinct mag\_code from entete\_ticket order by mag\_code;

-- Rajouter la ligne du total

-- Extraction table magasin au format .csv pour PowerBI

COPY (SELECT \* FROM magasin) To 'C:/Users/Public/Data\_Transverse/magasin.csv' With CSV DELIMITER ',' HEADER;

-- (B) Distance client magasin

-- [3] ETUDE PAR UNIVERS

-- (A) Histogramme de l'évolution du CA par univers entre 2016 et 2017

-- Où trouver la notion d'univers?

select \* from ref\_article;

-- CA en 2016 par code article

alter table ref\_article add CA\_2016 decimal(13,2);

update ref\_article set CA\_2016 = (select sum(tic\_totalttc) from entete\_ticket

join lignes\_ticket on entete\_ticket.idticket = lignes\_ticket.idticket

join ref\_article on ref\_article.codearticle = lignes\_ticket.idarticle

where extract(year from tic\_date) = 2016

group by ref\_article.codeunivers);

select sum(tic\_totalttc) from entete\_ticket

join lignes\_ticket on entete\_ticket.idticket = lignes\_ticket.idticket

join ref\_article on ref\_article.codearticle = lignes\_ticket.idarticle

where extract(year from tic\_date) = 2016

group by ref\_article.codeunivers;

-- CA en 2017 par code article

alter table ref\_article add ca\_2017 decimal(13,2);

update ref\_article set CA\_2017 = (select sum(tic\_totalttc) from entete\_ticket

join lignes\_ticket on entete\_ticket.idticket = lignes\_ticket.idticket

join ref\_article on ref\_article.codearticle = lignes\_ticket.idarticle

where extract(year from tic\_date) = 2017

group by ref\_article.codearticle);

-- On peut récupérer le CA depuis la table entete\_ticket avec la clé idticket commune à la table ligne\_ticket

-- Et l'associer à un code article commun aux tables ligne\_ticket et ref\_article

select ref\_article.codeunivers, sum(tic\_totalttc) as ca\_univers, extract(year from tic\_date) as année

from entete\_ticket

join lignes\_ticket on entete\_ticket.idticket = lignes\_ticket.idticket

join ref\_article on ref\_article.codearticle = lignes\_ticket.idarticle

where extract(year from tic\_date) between 2016 and 2017

group by ref\_article.codeunivers, année;

-- (B) Afficher le top 5 des familles les plus rentables par univers

select ref\_article.codeunivers, ref\_article.codefamille, lignes\_ticket.margesortie, EXTRACT (YEAR FROM tic\_date) AS année

FROM entete\_ticket JOIN lignes\_ticket on entete\_ticket.idticket = lignes\_ticket.idticket

JOIN ref\_article on ref\_article.codearticle = lignes\_ticket.idarticle

WHERE EXTRACT (YEAR FROM tic\_date) BETWEEN 2016 and 2017

GROUP BY ref\_article.codeunivers, ref\_article.codefamille, lignes\_ticket.margesortie, année

ORDER BY lignes\_ticket.margesortie DESC LIMIT 5;