TP 3

Les calculs

Rappel du contexte d'étude "magasin" L'entreprise que nous allons étudier est un commerce de proximité de type "supérette" organisé en rayons contenant des articles de diverses marques. Lors du passage en caisse, on conserve la date, le détail des articles achetés ainsi que le montant total qui a été réglé pour cet achat. On conserve également les informations concernant les fournisseurs et les livraisons. Pour chaque livraison, on mémorise le fournisseur ainsi que le ou les lots qui ont été livrés. Chaque lot concerne un seul article. Un lot est identifié par son numéro de lot et par le code-barres de l'article qu'il concerne. Un article possède un identifiant unique, le codebarres. C'est son appartenance à un lot qui permet de retrouver les informations qui le concernent (dates de fabrication, de livraison et de péremption). On peut d'ailleurs voir sur les photos suivantes quatre articles ayant le même codebarres, mais des numéros de lot différents et donc des dates de péremption différentes.

Les calculs simples

Le SQL nous permet d'appliquer les opérateurs d'addition, de soustraction, de multiplication et de division à des données de type numérique.
Par exemple, on veut faire une livraison et, pour cela, on a besoin de la liste desarticles mais on souhaite que le prix soit affiché avec un supplément de 1 € par article :

Ce qui nous donne, en SQL :

```
SELECT idArticle, libelle, prix + 1
FROM article
```

Testez la requête :

Nous savons maintenant utiliser les "alias" de colonnes. Prenons l'habitude de les utiliser dès que l'occasion se présente. Ici, par exemple, c'est une bonne occasion , en effet, dès lors que l'on utilise une opération, il faudra utiliser un alias de colonnes :

Ce qui nous donnes en SQL :

```
SELECT idArticle, libelle,
       prix + 1 AS "Prix livré"
FROM article
```

Testez la requête :

Autre exemple : finalement, on souhaite que le prix livré représente le prix des articles augmenté de 20 %.

Ce qui nous donnes en SQL :

```
SELECT idArticle, libelle, prix * 1.20 AS "Prix livré"
FROM article
```

Testez la requête :

Avec uniquement les calculs simples, les capacités de SQL sont vite limitées.
C'est pourquoi des fonctions permettant à SQL d'effectuer des opérations complexes ont été créées.

La fonction "COUNT"

La fonction "COUNT" vous permet d'obtenir le nombre de lignes que comporte la table résultante de votre requête.

-Par exemple, on veut obtenir le nombre d'articles présents dans notre base.

Ce qui nous donnes en SQL :

```
SELECT COUNT(*) AS "Nombre d'articles"
FROM article
```

Testez la requête :

Un autre exemple : on veut obtenir le nombre d'articles dont le prix est inférieurà 1 € :

```
SELECT COUNT(*) AS "Nombre d'articles à - d'1 euro"
FROM article
WHERE prix < 1
```

Testez la requête :

On souhaite obtenir le nombre de nos fournisseurs qui possèdent un email :

```
SELECT COUNT(*) AS "Nombre de fournisseurs avec email"
FROM fournisseur
WHERE email IS NOT NULL
```

En précisant un nom de colonne à la fonction COUNT, cela permet de ne prendre en compte que les lignes où cette colonne contient une valeur (non nulle). La requête suivante produit donc exactement le même résultat que la précédente :

```
SELECT COUNT(email)
       AS "Nb de fournisseurs avec email"
FROM fournisseur
```

Testez la requête :

On souhaite obtenir le nombre de fournisseurs qui nous ont déjà livrés. Il faut, pour cela, choisir la fonction "COUNT DISTINCT":

```
SELECT COUNT(DISTINCT idFournisseur)AS "Nb fourn."
FROM lot
```

Testez la requête :

Attention, si on oublie le DISTINCT, on obtient le nombre de lots que l'on nous a livrés, ce qui est complètement différent…

Les fonctions "MAX" et "MIN"

Les fonctions "MAX" et "MIN" retournent respectivement la valeur maximale et la valeur minimale qui se trouvent dans la colonne passée à la fonction.

-Par exemple, on veut obtenir la distance du fournisseur qui est le plus loin :

```
SELECT MAX(distanceKm) AS "Distance max fourn."
FROM fournisseur
```

Testez la requête :

Maintenant, on veut obtenir le prix de l'article le moins cher du rayon n° 1 :

```
SELECT MIN(prix) AS "Prix mini des art. du rayon n°1"
FROM article
WHERE idRayon = 1
```

Testez la requête :

Les fonctions "SUM" et "AVG"

Les fonctions "SUM" et "AVG" retournent respectivement la somme et la moyenne dela colonne passée à la fonction.
Par exemple, on veut obtenir le chiffre d'affaires réalisé le 1er avril 2010 : Ce qui nous donne, en SQL :

```
SELECT SUM(montantTotal) AS "CA le 01/04/2010"
FROM achat
WHERE dateAchat = '2010/04/01'
```

Testez la requête :

Maintenant, on veut obtenir le prix moyen d'un article du rayon n° 2 :

Ce qui nous donne, en SQL :

```
SELECT AVG(prix) AS "Moyenne des art. du rayon n°2"
FROM article
WHERE idRayon = 2
```

Testez la requête :

Chaque SGBDR dispose d'un large panel de fonctions qui n'est pas forcément compatible avec les autres SGBDR. Par exemple, dans PostgreSQL, pour extraire l'année d'une date, on utilise la fonction issue de la norme SQL:2003 ***EXTRACT(year from*** *champDate). Cette fonction est supportée par MySQL qui a également sa propre fonction* ***YEAR(****champDate****)*** *qui, elle, n'est pas supportée par PostgreSQL mais par SQL Server. Mais la fonction* ***NOW()*** *(non présente dans les différentes révisions de SQL), permet d'afficher la date et l'heure courante, existe sur quasiment tous les SGBDR.*

Exercice :

Réalisez les requêtes permettant d'obtenir les informations ci-dessous.

- La liste des articles ("idArticle", "libelle", "prix" que l'on renomme en "prix en €") avec en plus les prix en livres sterling (on prend comme taux de change :

1 £ = 1,21 €) ;

- Le nombre de marques référencées dans notre base de données ;
- Le chiffre d'affaires pour le mois d'avril 2010 ;
- La date de la dernière livraison reçue ;
- La distance moyenne des fournisseurs du Var (département 83) ;
- Le prix HT (taux de TVA de 20 %) de l'article le plus cher ;
- Le prix moyen au kilo des articles dont l'unité de mesure est le gramme.