# TP5 bis - Manipulation de données en SQL sous PostgreSQL

**Exercice 1 Le partitionnement** Cet exercice porte sur la base de donnés déjà exploitée dans le TP précédent. La clause *GROUP BY* d'une requête *SQL* permet de faire du partitionnement en regroupant les n-uplets suite à l'évaluation des clauses *FROM* et *WHERE*. Elle peut être complétée par une clause *HAVING* qui définit une condition de restriction ensembliste portant sur les parties obtenues. Les requêtes suivantes sont à écrire en utilisant la clause *GROUP BY*.

36. Par numéro article : la somme et le maximum des quantités commandées.
37. Pour chaque article, son nom, la plus petite quantité commandée, la plus grande et la moyenne des quantités commandées.
38. Le numéro des articles sujets à au moins 2 commandes.
39. Le numéro des articles sujets à au plus 3 commandes.
40. Le numéro des articles sujets à exactement 2 commandes.
41. Le nom des articles dont la somme des quantités commandées excède 100.
42. Le nom et le numéro des clients ayant passé au moins 2 commandes du même produit.
43. Le nom et le numéro des clients ayant passé au moins 2 commandes de produits différents.
44. La moyenne des quantités commandées de chaque article de type 'Litterature'.
45. La moyenne des quantités commandées pour les articles sujets à au moins 3 commandes.
46. Le numéro des articles dont la moyenne des quantités commandées est supérieure à 15.
47. Le nom des clients ayant effectué au moins 3 commandes.
48. Combien de jours séparent la première et la dernière commande de chaque client ?.
49. Quelle est la date de la dernière commande de chaque client ?.

**Exercice 2 Requêtes complexes**
Formulez en langage *SQL* les requêtes suivantes et exécutez-les en contrôlant la validité du résultat obtenu.

50. le nom des articles dont la quantité en stock est maximale.
51. le nom des articles dont au moins une commande est de quantité commandée supérieure à la quantité en stock.
52. le nom des articles dont la somme des quantités commandées est supérieure à la quantité en stock.
53. le nom des articles tels que toutes les commandes sont de quantité supérieure à celle en stock .
54. le numéro des articles qui ne sont commandés qu'une seule fois.
55. le numéro des clients qui ont commandé tous les livres de Philosophie.
56. le plus ancien client (numéro et nom).
57. le premier client à avoir acheté un exemplaire de '1984'.
58. la dernière commande en date de chaque client.

**Exercice 3 manipulation de dates**

Les requêtes que l'on demande de formuler dans cet exercice nécessitent l'usage de fonctions permettant de manipuler des expressions de type « date » dans l'environnement *Postgresql*.

Dans une base de donnes "PostgreSQL", une valeur de type « Date » exprime une date du calendrier, une valeur de type « timestamp » représente l'instant (date + heure) avec une précision de l'ordre de la microseconde alors qu'une valeur de type « time » détermine une heure à la microseconde près. Un ensemble de fonctions est mis à disposition des usagers, permettant l'accès aux éléments constitutifs d'une date (jour, mois, année, heure, minute, secondes...) selon le type employé.

Parmi ces fonctions, on peut citer :

```
to_char(une_date, modèle de format)
to_date(une_chaine,modèle de format)
```

qui permettent de convertir des données de type date au format chaîne de caractères et vice versa. Les modèles de format sont décrits dans la documentation dans la rubrique *fonctions de formatage*.
Ou encore :

```
extract(champ from timestamp)
```

qui permet l'accès aux champs constitutifs d'une date.

Une liste exhaustive de ces fonctions est disponible dans la documentation dans la rubrique *fonctions de traitement de la date et de l'heure*

**Exemple d'utilisation de ces fonctions :**

Afficher la date des commandes passées par le client numéro 6 selon le format "Mardi 24 janvier 2022".

```
SELECT  to_char(date_com, 'day dd month yyyy')
FROM commandes
WHERE num_cli_c = 6;
```

...En Français dans le texte :

```
SELECT  to_char(date_com, 'tmday dd tmmonth yyyy')
FROM commandes
WHERE num_cli_c = 6;
```

Si l'on préfère la forme "31/03/2022", la requête deviendra :

```
SELECT  to_char(date_com, 'dd/mm/yyyy')
FROM commandes
WHERE num_cli_c = 6;
```

Insérer dans la table "commandes" la commande de 25 unités de l'article numéro 3 par le client numéro 6 passée le Mardi 24 janvier 2022.

```
INSERT INTO commandes
VALUES ( to_date('mardi 24 janvier 2012','day dd month yyyy'), 3, 6, 25);
```

Opérations sur les données de type "date" (liste non exhaustive).

# Travail à réaliser :

Formulez les requêtes suivantes en langage SQL :

59. La date d'aujourd'hui selon le format : « lundi 16 décembre 2013 ».
60. L'heure courante sous la forme : « 13 :24 :52 ».
61. Quelles sont les commandes passées en octobre 2013 ?
62. Quelles sont les commandes passées il y a au plus trente jours ?
63. Nom des clients ayant passé des commandes il y a au plus trente jours ?
64. Quelles sont les commandes passes entre le 16/08/2013 et le 16/11/2013 ?
65. Nom des clients ayant passé une commande un vendredi ?
66. Nom des clients ayant passé une commande un jeudi ou un vendredi ?
67. Quels sont les clients (numéro et nom) qui ont passé des commandes au cours du premier trimestre de l'année courante ?
68. Quels sont les clients (numéro et nom) qui n'ont pas passé de commande durant les deux derniers mois (60 jours) ?
69. Quels sont les clients qui ont passé toutes leurs commandes durant le même mois ?