Exercice 1 : Finir l'exercice 3 du précédent TP (requêtes 11, 12 et 13) et réorganiser le travail à faire selon les fichiers suivants :
- `creation.sql` contenant la définition du schéma de la base de données relationnelle en prenant en compte ses contraintes. Afin de pouvoir travailler en groupe lors du prochain TP et échanger les données avec les membres de votre groupe, respecter les noms des relations et attributs de l'énoncé,
- `insertion.sql` contenant les n-uplets,
- `affichage.sql` permettant d'afficher le contenu des tables créées,
- `requete.sql` pour les requêtes qui auront du être testées une à une,
- `nettoyage.sql` dont l'exécution permet de détruire toutes les tables que vous avez créés.

Testez vos requêtes en utilisant le fichier `tp3_ex0.sql` (voir sur mon compte : /export/home/users/Enseignants/mogbi
).

Exercice 2 : Une histoire de livres

On considère la base consituée des tables suivantes :

```
LIVRES(Code, Titre, Auteurs, Nom_ed, Annee_ed)
PRETS(Code, #carte, Date_emprunt)
MEMBRES(#carte, Nom, Adr, Code_postal,Ville)
EDITEURS(Nom_ed, Adr, Code_postal,Ville)
ACQU(Code, Date_acqu, Prix_acqu)
EMPLACEMENTS(Code, Site)
```

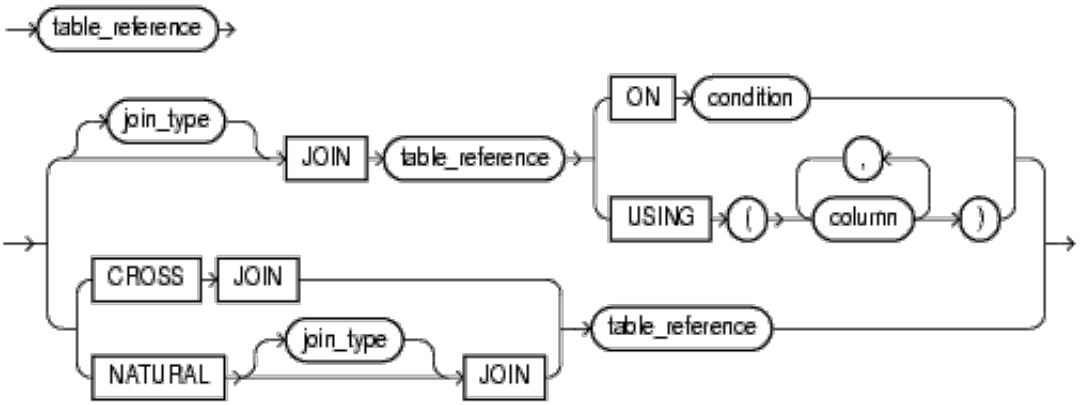
Exprimez en SQL les requêtes suivantes (si possible avec une seule une commande) :

1. (a) Quel est le nombre total de livres de la bibliothèque ?
   (b) Quel est le nombre de livres de la bibliothèque, parus en 2004 ?
   (c) Quel est le nombre de livres de la bibliothèque par éditeur ?
   (d) Pour chaque livre de la bibliothèque (Titre,Auteur) quel est le nombre de ses éxemplaires ?
   (e) Quel est le nombre de livres sur chaque site ?
2. (a) Quel est le total des prix de tous les livres ?
   (b) Quelle est la somme des prix des livres achetés chez chaque éditeur ?
   (c) Quelle est la somme des prix des livres achetés chez chaque éditeur parisien ?
   (d) Quelle est la somme des prix des livres achetés chez chaque éditeur ayant vendu plus de 5 livres à la bibliothèque ?
   (e) La personne de numéro de carte 123456 a perdu tous ses livres et elle doit rembourser la bibliothèque 80% du prix d'acquisition de chaque livre, combien a-t-elle à rembourser ?
3. (a) Quel est le code et le prix de l'exemplaire (des exemplaires) le plus cher (les plus chers) ?
   avec (>= ALL mais pas de max) ou (avec IN à la place de =)
   (b) Quel est le prix moyen des livres de la bibliothèque ?
   (c) Quel est l'auteur, le titre et le prix du livre (Titre, Auteur) le plus cher de la bibliothèque ?
   (d) Pour chaque auteur, quel est son livre le plus cher à la bibliothèque ?
   (e) Quel est le livre (Titre, Auteur) le plus cher de chaque site ?

4. (a) Quels sont les livres (Titre, Auteur) dont la bibliothèque possède plus de 10 exemplaires ?
   (b) Quel est le site qui a le plus grand nombre de livres ?
   (c) Quel est l'éditeur dont la bibliothèque possède le plus grand nombre de livres ?
   (d) Quel est l'auteur (le groupe d'auteurs) dont la bibliothèque possède le plus grand nombre de livres ?
   (e) Parmis les éditeurs qui ont vendu plus de 50 livres à la bibliothèque quel est celui qui a reçu la plus grande somme pour l'achat des livres ?

Exercice 3 : `JOIN`

Voici quelques indications sur la jointure sous `ORACLE` (dans le champs `FROM`) où `table_reference` désigne un nom de table :



avec `join_type : :=` { `INNER` | { `LEFT|RIGHT|FULL` } `[OUTER]` }

Les jointures sont respectivement la θ-jointure (`ON condition`), l'équi-jointure (`USING (column)`), le produit catésien (`CROSS`) et la jointure naturelle (`NATURAL`). Le type dans la jointure naturelle désigne respectivement les n-uplet joints `INNER` (valeur par défaut), les n-uplet joints et ceux de la table de gauche/droite ou toutes (resp. `LEFT/RIGHT ou FULL`). Remarquez qu'il peut-être nécéssaire de paranthèser les jointure.
Reformuler si possible les requêtes de l'exercice précédent en utilisant `JOIN`.

Exercice 4 : pour ceux qui en veulent plus... Pour cette partie, vous réutiliserez les tables du TP précédent avec des dates de commandes.
Exprimez les requêtes SQL suivantes :

1. Donner pour chaque client ayant des commandes en cours, le nom le prénom et les dépenses engendrées par l'ensemble de ses commandes.
2. Donner le nom et le prénom des clients qui commandent au moins un des produits commandé par Mr Barnabe Laurent. (Chaque nom,prenom ne doit apparaître qu'une seule fois)
3. Donner la raison sociale et les recettes en cours de tous les fournisseurs de la famille Dupont.
4. Donner les noms et prénoms des clients qui n'ont commandé que des souris (On suppose que toutes les désignations de souris contiennent la chaine 'souris').
5. Donner les noms et prénoms des clients qui n'ont commandé que des souris, ou n'ont rien commandé du tout.
6. Donner les désignations des produits qui n'ont été commandés que par un seul client.
7. Donner les noms et prénoms des clients qui ont commandé moins de 40 unités d'articles au total.
8. Donner les clients qui ont commandé tous les produits. – *Commande / Produit*
   – *quels sont les clients pour lesquels il n'y a pas de produits qu'ils n'ont pas commandé.*

Annexe pour le TP : la clause `SELECT`

- `SELECT [ALL|DISTINCT] colonne | (fct(colonne) [AS nom]) [, liste_col_ou_fct(col)]`
  Liste de colonne ou fonctions sur une colonne (`AVG|MAX|MIN|SUM [ALL|DISTINCT]`) ou sur le nombre de ligne d'une table (`COUNT [ALL|DISTINCT]`). S'il y a des regroupements de colonnes `GROUP BY` alors les fonctions se font sur chaque regroupement (`AVG, MAX, MIN`) ou sur le nombre de ligne de chaque regroupement (`COUNT`).

- `FROM nom_de_table [alias] [,liste_de_nom_de_table]`
  Indique les tables à partir desquelles récupérer les données. Un alias permet un renommage local. On utilise l'alias à la place du nom de la table afin d'enlever des ambiguités, de simplifier l'écriture ou d'être plus explicite.

- `[WHERE condition]`

  Exemples d'opérateurs utilisables dans une condition :
  - opérateurs logiques (`AND, OR, NOT, IS NULL, IS NOT NULL`)
  - opérateurs de chaines (`[NOT] IN|BETWEEN|LIKE`)
  - opérateurs arithmétiques (`+, -, *, /, mod(,)`)
  - comparateurs arithmétiques (`<, >, =, <>, <=, >=, [NOT] IN`)

  Conditions avec sous-requêtes :
  ```
  WHERE [NOT] EXISTS|IN (clause_select)
  WHERE colonne | (fct(colonne) [AS nom]) op [ALL|ANY] (clause_select)
  ```
  avec
  `ALL` : la condition est vraie seulement si elle est vraie pour toutes les valeurs de la clause select. Donc la condition est vraie si la sous-requête est vide. Donc : `<ALL` = moins que le minimum, `>ALL` = plus que le maximum, `<>ALL` equivalent à `NOT IN`.
  `ANY` : la condition est vraie seulement si elle est vraie pour au moins l'une les valeurs de la clause select. Donc : `<ANY` = moins que le maximum, `>ANY` = plus que le minimum, `=ANY` equivalent à `IN`.

- `GROUP BY colonne [,list_col]`
  Regroupe des résultats selon une(des) colonne(s).

- `HAVING condition`
  est à `GROUP BY` ce que `WHERE` est à `FROM`.

- `[ORDER BY colonne [ASC|DESC] [,list_col]]`
  Tri des résultats selon une (des) colonne(s). Toutes colonnes du select qui n'est pas sous une fonction doit etre spécifiée dans la liste des colonnes de `ORDER BY` si on souhaite effectuer un tri.

Clauses obtenues par opération emsembliste :
```
(clause_select) UNION [ALL] (clause_select)
(clause_select) INTERSECT [ALL] (clause_select)
(clause_select) MINUS [ALL] (clause_select)
```
Sans l'option `ALL` seuls les n-uplets distinct sont conservés (à tester : support de `ALL` sous Oracle 9i pour les deux dernières opérations). Rappel : il faut respecter les types (i.e. les domaines en algèbre relationnelle) colonne à colonne.