Bases de données - TP 3 (2 séances)

Code de partage avec Capytale : 49cb-9154632

On commencera par exécuter les requêtes permettant de créer et remplir les deux tables.

Structure de la base de donnée du TP

Dans ce TP, la base de données contient deux tables :

La table de véhicules nommée vehicules. Les colonnes de cette table sont :
- l'identifiant id_vehicule , qui peut correspondre à un numéro de série, de type integer, ce champ correspondra à une clef primaire (primary key)
- la marque **marque**, de type **text**
- le pays de fabrication **pays**, de type **text**
- le modèle **modele**, de type **text**
- le type de motorisation **moteur**, de type **text**
- la consommation moyenne en ville **cons_ville**, de type **float**
- la consommation moyenne sur autoroute **cons_autoroute**, de type **float**
- le prix du véhicule neuf **prix_neuf**, de type **integer**

Une table nommée annonces correspondant aux annonces de ventes de voitures d'occasion pour un revendeur, et contenant dans chaque ligne :
- le numéro de l'annonce **id_annonce**, de type **integer**, ce champ correspondra à une clef primaire (**primary key**)
- l'identifiant du véhicule **id_vehicule**, de type **integer**, ce champ correspondra à une clef étrangère (**foreign key**) (utile pour les jointures)
- l'année de mise en circulation **annee**, de type **integer**
- le kilométrage **km** , de type **integer**
- le prix de vente **prix**, de type **integer**

Obtenir les différents champs et différentes tables possibles

La commande **SELECT DISTINCT** *colonne* **FROM** *table* renvoie les différents champs rencontrés dans la colonne mentionnée de la table indiquée.

On peut tester avec l'attribut `pays`

```
select distinct pays from vehicules
```

Exercices

Exercice 1

1. Chercher dans la table vehicules toutes les berlines électriques japonaises.

   Il suffit de préciser les conditions :

```
select * from vehicules
where modele="Berline" and pays="JPN" and moteur="Electrique"
```

2. Chercher dans la table vehicules toutes les citadines dont le prix est supérieur à 20 000 euros.

```
select * from vehicules where modele="Citadine" and prix_neuf>=20000
```

3. Chercher dans la table vehicules toutes les citadines européennes à moteur thermique dont le prix est supérieur à 20 000 euros.

```
select * from vehicules
where modele="Citadine"
and (pays="GER" or pays="FRA" or pays="SWE" or pays="ITA")
and (moteur="Essence" or moteur="Diesel")
and prix_neuf>=20000
```

Exercice 2

Dire ce que la commande suivante affiche :
SELECT *marque, modele, moteur, (cons_autoroute+cons_ville)/2* **FROM** *vehicules*

Cette commande donne une consommation moyenne (entre celle en ville et celle sur autoroute) pour chaque véhicule.

Exercice 3

On rappelle que les opérations arithmétiques usuelles pour les nombres : +, -, *, /, MIN, MAX,... sont naturelles.

1. Afficher la consommation maximum sur autoroute des berlines.

```
select max(cons_Autoroute) from vehicules where modele="Berline"
```

On trouve 7,9

2. Afficher la consommation minimum en ville des berlines européennes.

```
select min(cons_Ville) from vehicules
where modele="Berline"
and (pays="GER" or pays="FRA" or pays="SWE" or pays="ITA")
```

On trouve 7,1

Exercice 4 - jointure

On rappelle la commande générale pour une jointure entre deux tables :

SELECT *col1, col2...* **FROM** *table1* **INNER JOIN** *table2* **ON** *condition*

Dans *condition*, les tables 1 et 2 doivent être liées par exemple par un champ d'une clef étrangère pour la table 1 égal à un champ pour une clef primaire pour la table 2.

Dans la table annonce, trouver tous les modèles avec les marques de véhicules électriques dont le prix de vente est inférieur à 25 000 euros.

On effectue la jointure sur l'attribut `id_vehicule` qui est présent dans les deux tables (d'où la précision du nom de la table dans la requête.

```
select modele, marque
from vehicules
inner join annonce on vehicules.id_vehicule=annonce.id_vehicule
where moteur="Electrique" and prix<=25000
```

Exercice 5 - order by

Trouver la liste des modèles et marques de voitures consommant moins de 5 litres aux 100 dans la table vehicules. Les classer par prix du neuf.

Le type de consommation n'est pas précisé, donc on calcule la consommation moyenne (on voit que ce type de calcul peut être intégré dans la condition).

```
select marque, modele from vehicules
where (cons_Autoroute+cons_Ville)/2<5 order by prix_neuf
```

Exercice 6 - commande sum

La commande **SUM** permet de faire une somme d'éléments sur une colonne par exemple, la commande :

SELECT SUM(*col*) **from** *table*
renvoie la somme de la colonne *col* de la table *table*.

Déterminer la valeur vénale (=somme des prix affichés) des annonces de la table annonce.

```
select sum(prix) from annonce
```

On trouve 2 281 189

Exercice 7

La commande **COUNT** compte le nombre d'éléments dans une colonne par exemple, la commande

SELECT COUNT(*col*) **FROM** *nom de table* [**WHERE** *condition*]
ou, sans spécifier de colonne :

SELECT COUNT(*) **FROM** *nom de table* [**WHERE** *condition*]
Les crochets signifient que la commande **WHERE** est optionnelle (pas de crochets quand vous l'utilisez !).

1. Compter le nombre de voitures électriques dans la table vehicules

   ```
   select count(*) from vehicules where moteur="Electrique"
   ```

   On trouve 17

2. Compter le nombre d'enregistrements dans la colonne **cons_autoroute**.

   ```
   select count(cons_autoroute) from vehicules
   ```

   On trouve 83, ce qui correspond en fait aux 100 véhicules de la table moins les 17 véhicules électriques (de la question précédente).

3. Compter le nombre de modèles avec un moteur thermique qui affichent une consommation en ville inférieure à 5L/100km.

   ```
   select count(modele) from vehicules
   where (moteur="Diesel" or moteur="Essence") and cons_ville<=5
   ```

On trouve 4

Exercice 8 - commande group by

La commande **GROUP BY** permet de regrouper les données suivant une colonne par exemple en évitant les doublons. Avec notre table vehicules :

SELECT COUNT(*), *pays* **FROM** *vehicules* **GROUP BY** *pays* renvoie le nombre de vehicules par pays des voitures proposées.

Afficher par pays le nombre de voitures électriques disponibles dans la table vehicules. Sur la base du modèle, avec une condition supplémentaire :

```
SELECT pays, COUNT(*) FROM vehicules where moteur="Electrique" GROUP BY pays
```

Exercice 9 - tables et jointures

1. Créer une (ou plusieurs) table(s) pour aboutir à une table contenant deux colonnes : une première colonne donnant l'âge du véhicule, par ordre croissant, et une deuxième colonne donnant le nombre de véhicules à un prix inférieur à 20 000 euros.
   Le prix considéré sera celui de la table annonce (occasion).

   On va faire la sélection de données dans un premier temps, on renomme les noms de colonnes pour faciliter la compréhension et pour pouvoir appliquer les commande `group by` et `order by` (cette dernière n'est en fait pas indispensable car le `group by` crée une sélection ordonnée).

   ```
   select 2026-annee as age, count(prix) as nombre_vehicules from annonce
   where prix<=20000 group by  age order by age
   ```

   Ensuite il suffit de faire un `create table as` puis d'ajouter le résultat de la requête précédente :

   ```
   create table extrait_annonce as
   select 2026-annee as age, count(prix) as nombre_vehicules
   from annonce
   where prix<=20000 group by  age order by age
   ```

2. Même question en ajoutant une colonne donnant le nombre de pays d'origine différents pour chaque âge de voiture.

   On doit faire une jointure pour obtenir l'information sur les pays, on doit afficher en plus cette fois la colonne donnant le nombre de pays d'origine différents pour chaque âge de voiture, d'où le `distinct` :

   ```
   create table extrait_2 as
   select 2026-annee as age, count(prix) as nombre_vehicules, count(distinct
      pays) as nombre_pays
   from annonce
   inner join vehicules on annonce.id_vehicule=vehicules.id_vehicule
   where prix<=20000 group by age order by age
   ```

Exercice 10 - concaténation de chaînes de caractères : la commande ||

La commande || permet de concaténer deux champs de chaînes de caractères.

SELECT *col1 || col2 || col3 ...* **FROM** *table*

ou :

SELECT *col1 || chaîne de caracteres || col2 || col3 ...* **FROM** *table*

Dans certains SGBD, la commande **CONCAT()** peut être utilisée à la place.

Compléter la commande suivante permettant de renvoyer le modèle accolé à la motorisation des voitures de la table vehicules :
SELECT ..... **FROM** *vehicules*
On peut proposer par exemple la requête suivante qui garde toutes les informations en accolant le modèle et la motorisation :

```
select id_vehicule, marque, modele || " - " || moteur, prix_neuf from vehicules
```