Secteur Tertiaire Informatique
Filière « Etude et développement »

Séquence « Développer des pages Web »

SQL & MySQL

Apprentissage

Mise en pratique

Evaluation

1. Exercices de SQL sur la Base Magasin

En utilisant les scripts de création de la base **Magasin** qui vous ont été fourni, vous devrez créer en local votre Base Magasin.

Ensuite, vous devrez répondre aux questions suivantes :

Question 1

Dans la base magasin, sélectionnez les articles dont le prix est inférieur à 1 500 euros.

```
SELECT * FROM `article` WHERE prix < 1500
```

Question 2

Dans la base magasin, sélectionner les clients qui habitent à Paris

```
SELECT nom,prenom,adresse,mail FROM client WHERE ville='Paris'
```

Question 3

Dans la base magasin, sélectionner les clients qui ont un âge supérieur à 40 ans.

```
SELECT nom, prenom, age, adresse, ville FROM client WHERE age >40
```

Question 4

Dans la base magasin, sélectionner les clients qui n'ont pas d'adresse email

```
SELECT nom, prenom, age, adresse, ville FROM client WHERE mail IS NULL
```

Question 5

Dans la base magasin, sélectionner les clients dont le nom commence par une lettre comprise entre A et H, la lettre H étant non comprise.

```
SELECT nom, prenom, age,ville FROM client WHERE nom BETWEEN 'A' AND 'H'
```

Question 6

Dans la base magasin, sélectionner les clients dont l'âge figure dans la liste (18,19,20)

```
SELECT nom, prenom, age, ville FROM client WHERE age IN ( 18, 19, 20)
```

Question 7

Dans la base magasin, sélectionner les clients qui habitent une avenue, et dont l'adresse contient donc la chaîne de caractères « Av » (sensible à la casse).

```
SELECT nom, prenom, age, adresse, ville FROM client WHERE adresse LIKE '
%Av%'
```

Question 8

Dans la base magasin, sélectionnez les articles dont le prix est compris entre 100 et 500 euros.

```
SELECT * FROM article WHERE prix BETWEEN 100 AND 500
```

Question 9

Dans la base magasin, sélectionnez tous les articles de marque Nikon (dont la désignation contient ce mot).

```
SELECT * FROM article WHERE designation LIKE '%Nikon%'
```

Question 10

Dans la base magasin, sélectionnez tous les caméscopes, leur prix et leur référence.

```
SELECT id_article, prix FROM article WHERE designation LIKE '%Camescope%'
```

Question 11

Dans la base magasin, sélectionner les clients dont le nom commence par **Ma** à l'aide d'une *expression régulière*.

```
SELECT nom, prenom, age, adresse, ville FROM client WHERE nom REGEXP 'Ma.*'
```

Question 12

Dans la base magasin, sélectionner les clients qui ont moins de 30 ans et qui habitent à Paris.

```
SELECT nom, prenom, age, adresse,ville FROM client WHERE age <30 AND ville = 'Paris'
```

Question 13

Dans la base magasin, sélectionner les clients qui habitent à **Lyon** ou dont le nom commence par la lettre **H**.

```
SELECT nom, prenom, age, adresse,ville FROM client WHERE ville='Lyon' OR nom LIKE 'H%'
```

Question 14

Dans la base magasin, sélectionner les clients qui n'habitent ni à **Lyon** ni à **Paris**.

```
SELECT nom, prenom, age, adresse,ville FROM client WHERE NOT(ville='Paris') AND NOT(ville='Ly
on')
```

Question 15

Dans la base magasin, calculer l'âge moyen des clients

```
SELECT AVG(age) FROM client
```

Question 16

Dans la base magasin, calculer le nombre de villes différentes de la table client.

```
SELECT COUNT(DISTINCT ville) FROM client
```

Question 17

Dans la base magasin, sélectionnez tous les produits de la catégorie informatique, et affichez leur code, leur désignation et leur prix par ordre décroissant de prix.

```
SELECT id_article, designation, prix FROM article WHERE categorie LIKE '%informatique%' ORDER
 BY prix DESC
```

Question 18

Dans la base magasin, sélectionnez tous les clients de moins de 40 ans, et ordonnez les résultats par ville en ordre alphabétique.

```
SELECT nom, prenom, age, adresse,ville FROM client WHERE age < 40 ORDER BY ville
```

Question 19

Dans la base magasin, calculez le prix moyen de tous les articles.

```
SELECT AVG(prix) FROM article
```

Question 20

Dans la base magasin, calculez le nombre d'e-mails non NULL et distincts l'un de l'autre.

```
SELECT DISTINCT COUNT(mail) FROM `client` WHERE mail IS NOT null
```

Question 21

Dans la base magasin, déterminer l'âge du client le plus jeune.

```
SELECT MIN(age) FROM client
```

Question 22

Dans la base magasin, calculer le nombre total des articles commandés

```
SELECT SUM(quantite) FROM ligne
```

Question 23

Dans la base magasin, calculer l'âge moyen des clients par ville.

```
SELECT AVG( age ) AS 'age moyen', ville FROM client GROUP BY ville
```

Question 24

Dans la base magasin, afficher l'âge minimum, l'âge maximum et l'âge moyen des clients par ville.

```
SELECT MIN(age) AS 'Age minimum',AVG(age) AS 'Age moyen', MAX(age) AS 'Age maximum', ville FROM client GROUP BY ville
```

Question 25

Dans la base magasin, calculer l'âge moyen des clients qui habitent une ville dont le nom commence par un **L**

```
SELECT AVG( age ),ville FROM client GROUP BY ville HAVING ville LIKE 'L%'
```

Question 26

Dans la base magasin, sélectionnez tous les articles commandés par chaque client.

```
SELECT nom,prenom,article.id_article,designation FROM client,commande,article,ligne WHERE client.id_client=commande.id_client AND ligne.id_comm=commande.id_comm AND ligne.id_article=article.id_article
```

Le même résultat de requête formulé avec une jointure interne JOIN … ON :

```
SELECT article.designation, client.nom, client.prenom FROM `article` JOIN Ligne on article.id_article = ligne.id_article JOIN commande on ligne.id_comm = commande.id_comm JOIN Client on commande.id_client = client.id_client
```

Question 27

Dans la base magasin, sélectionnez tous les clients dont le montant d'une commande dépasse 1 500 euros.

```
SELECT client.id_client,nom,prenom, ligne.id_comm, sum(prix_unit*quantite) AS 'total' FROM client,ligne,commande WHERE ligne.id_comm=commande.id_comm AND commande.id_client=client.id_client GROUP BY ligne.id_comm HAVING total>1500
```

```
SELECT client.id_client, client.nom, client.prenom, ligne.id_comm, sum(prix_unit*quantite) as "total" FROM client INNER JOIN commande on client.id_client = commande.id_client INNER JOIN ligne on commande.id_comm = ligne.id_comm GROUP BY ligne.id_comm HAVING total>1500
```

Question 28

Dans la base magasin, sélectionnez tous les clients dont le montant total de toutes les commandes dépasse 5 000 euros.

```
SELECT client.id_client, client.nom, ligne.id_comm, sum(prix_unit*quantite) FROM client,ligne,commande WHERE ligne.id_comm=commande.id_comm AND commande.id_client=client.id_client GROUP BY client.id_client HAVING sum(prix_unit*quantite)>5000
```

```
SELECT client.id_client, client.nom, ligne.id_comm, sum(prix_unit*quantite) as "total" FROM client INNER JOIN commande ON client.id_client = commande.id_client INNER JOIN ligne ON commande.id_comm = ligne.id_comm GROUP BY client.id_client HAVING total > 5000
```

```
Ou avec une Sous-requête :
SELECT nom, id, ROUND(mtn,2) FROM (SELECT a.nom AS nom, b.id_client AS id, SUM(c.prix_unit*c.quantite) AS mtn FROM client a, commande b, ligne c WHERE b.id_comm = c.id_comm and b.id_client = a.id_client GROUP BY nom, id) d WHERE mtn > 5000
```

Question 29

Dans la base magasin, à l'aide d'une jointure, retrouver toutes les commandes faites par un client. (Donner les deux écritures possibles de cette requête, celle qui utilise WHERE et celle qui utilise INNER JOIN)

```
SELECT commande.id_comm,nom,prenom,ville FROM commande,client WHERE client.id_client = commande.id_client ORDER BY nom
```

```
SELECT commande.id_comm,nom,prenom,ville FROM commande INNER JOIN client ON client.id_client = commande.id_client ORDER BY nom
```

Question 30

Dans la base magasin, à l'aide d'une jointure, afficher la liste des articles les plus vendus sur le site et à afficher leur code, leur désignation, leur prix et le nombre total d'articles. (Donner les deux écritures possibles de cette requête, celle qui utilise WHERE et celle qui utilise INNER JOIN)

```
SELECT article.id_article, designation, prix, SUM( quantite ) AS 'Total' FROM article, ligne WHERE article.id_article = ligne.id_article GROUP BY id_article ORDER BY Total DESC
```

```
SELECT article.id_article, designation, prix, SUM( quantite ) AS 'Total' FROM article INNER JOIN ligne ON article.id_article = ligne.id_article GROUP BY id_article ORDER BY Total DESC
```

Question 31

Dans la base **magasin** et dans la table **client** ajouter un champ **code postal** après le champ **ville**.

Vous avez donc maintenant un nouveau champ code postal dont les valeurs sont nulles pour chaque client.

Créer une contrainte sur le **Code Postal Client** pour que, s'il est non renseigné il soit par défaut fixé à '19240' (Allassac) ou bien compris entre '01000' et '99999'. Vous devrez créer pour cela un **trigger** qui enverra un message d'erreur lorsque cette contrainte ne sera pas respectée avant l'insertion et la modification d'un enregistrement.

Vous pourrez vous référer à ce lien :

https://dev.mysql.com/doc/refman/8.0/en/signal.html

… et à la documentation fournie pour en apprendre davantage sur le concept de **trigger** de **MySQL**

Faire un essai avec par exemple :

```
INSERT INTO `client`(`nom`, `prenom`, `age`, `adresse`, `ville`, `code_postal`, `mail`) VALUES ("RESTOUEIX","Sacha",52,"25 Route de la Plaine","Allassac","00999","sacha8milo@gmail.com")
```

```
INSERT INTO `client`(`nom`, `prenom`, `age`, `adresse`, `ville`, `code_postal`, `mail`) VALUES ("RESTOUEIX","Sacha",52,"25 Route de la Plaine","Allassac","92300","sacha8milo@gmail.com")
```

```
DROP TRIGGER IF EXISTS `before_insert_cp`;
DELIMITER $$
CREATE TRIGGER `before_insert_cp` BEFORE INSERT ON `client` FOR EACH ROW
BEGIN
 IF NEW.code_postal IS NOT NULL   -- le code postal est NON NULL
    AND NEW.code_postal NOT BETWEEN "01000" AND "99999"
      THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Mauvais Code
Postal';
    END IF;
END
$$
DELIMITER ;
```

Question 32

Vous allez ajouter un champs **version** de type **int** à la table **client** de la base **magasin**. Avec par défaut la valeur 0.

Vous allez créer une nouvelle table **historique_client** qui aura la structure suivante :

```
CREATE TABLE historique_client (
  id BIGINT NOT NULL AUTO_INCREMENT,
  action ENUM ( 'update' , 'delete' ) DEFAULT NULL,
  date_action DATETIME DEFAULT NULL,
  version BIGINT NOT NULL DEFAULT 0,
  id_original BIGINT NOT NULL DEFAULT 0,
  nom VARCHAR(50) DEFAULT NULL,
  PRIMARY KEY (id),
  KEY (id_original),
  KEY (action),
  KEY (date_action),
  KEY (version)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Vous allez devoir écrire un **Trigger** qui avant chaque **modification** :

– A chaque **modification** d'un enregistrement de la table « **client** », il va recopier les données dans la table « **historique_client** » et gérer les champs qui servent à l'historique. C'est à dire incrémenter le numéro de **version**, renseigner le champ « **action** » à la valeur « **update** » et le champ « **date_action** » à la date et heure courante.

```
BEGIN
    SET NEW.version = OLD.version + 1;
    INSERT INTO historique_client
      (action, date_action, version, nom, id_original)
    VALUES
      ('update', NOW(), OLD.version, OLD.nom, OLD.id_client);
END
```

Vous ferez de même après chaque **suppression** d'un **client** :

– A chaque **suppression** d'un enregistrement de la table « **client** », il va recopier les données (qui vont être supprimées dans la table d'origine) et là encore gérer les champs qui servent à l'historique. C'est à dire incrémenter le numéro de **version**, renseigner le champ « **action** » cette fois-ci avec la valeur « **delete** » et le champ « **date_action** » à la date et heure courante.

```
BEGIN
    INSERT INTO historique_client
      (action, date_action, version, nom, id_original)
    VALUES
      ('delete', NOW(), OLD.version, OLD.nom, OLD.id_client);
END
```

Vous pourrez tester ensuite qu'à chaque modification d'un client, la table **historique_client** sera alimentée.

Pour la **suppression**, vous ne pourrez bien entendu pas supprimer des clients qui sont engagés dans une **commande** sur le principe de l'intégrité des données portant sur les **clefs étrangères**.