Exercice SQL sur les villes et départements de France

Analyse exploratoire

Affichage de la table

```
SELECT * FROM villes_france_free ;
SELECT * FROM departement ;
```

Pour récupérer les noms des colonnes de la base

```
SELECT COLUMN_NAME
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'villes_france_free';
```

Obtenir la liste des 10 villes les plus peuplées en 2012

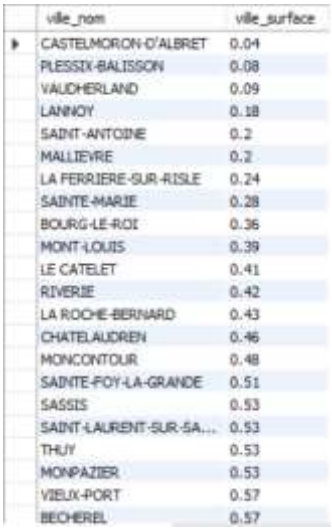
 # Utilisation de la table 'villes_france_free'

```
SELECT ville_nom, ville_population_2012
FROM villes_france_free
ORDER BY ville_population_2012 DESC
LIMIT 10;
```

ville_nom	ville_population_2012
PARIS	2211000
MARSEILLE	851400
LYON	474900
TOULOUSE	439600
NICE	344900
NANTES	283300
STRASBOURG	272100
MONTPELLIER	253000
BORDEAUX	235900
LILLE	225800

Obtenir la liste des 50 villes ayant la plus faible superficie

 # Utilisation de la table 'villes_france_free'



ville_nom	ville_surface
CASTELMORON-D'ALBRET	0.04
PLESSIX-BALISSON	0.08
VAUDHERLAND	0.09
LANNOY	0.18
SAINT-ANTOINE	0.2
MALLIEVRE	0.2
LA FERRIERE-SUR-RISLE	0.24
SAINTE-MARIE	0.28
BOURG-LE-ROI	0.36
MONT-LOUIS	0.39
LE CATELET	0.41
RIVERIE	0.42
LA ROCHE-BERNARD	0.43
CHATELAUDREN	0.46
MONCONTOUR	0.48
SAINTE-FOY-LA-GRANDE	0.51
SASSIS	0.53
SAINT-LAURENT-SUR-SA...	0.53
THUY	0.53
MONPAZIER	0.53
VIEUX-PORT	0.57
BECHEREL	0.57

```
SELECT ville_nom, ville_surface
FROM villes_france_free
ORDER BY ville_surface ASC
LIMIT 50;
```

Obtenir la liste des départements d'outres-mer, c'est-à-dire ceux dont le numéro de département commencent par "97"

 # Utilisation de la table departement

departement_nom	departement_code
Guadeloupe	971
Martinique	972
Guyane	973
Réunion	974
Mayotte	976

```
SELECT departement_nom, departement_code
FROM departement
WHERE departement_code LIKE "97%";
```

Obtenir le nom des 10 villes les plus peuplées en 2012, ainsi que le nom du département associé

ville_nom	departement_nom
PARIS	Paris
MARSEILLE	Bouches-du-Rhône
LYON	Rhône
TOULOUSE	Haute-Garonne
NICE	Alpes-Maritimes
NANTES	Loire-Atlantique
STRASBOURG	Bas-Rhin
MONTPELLIER	Hérault
BORDEAUX	Gironde
LILLE	Nord

```
SELECT ville_nom, departement_nom
FROM villes_france_free AS V
RIGHT JOIN departement AS D ON ville_departement =
departement_code
ORDER BY V.ville_population_2012 DESC
LIMIT 10;
```

Obtenir la liste du nom de chaque département, associé à son code et du nombre de commune au sein de ces département, en triant afin d'obtenir en priorité les départements qui possèdent le plus de communes

```
SELECT departement_nom, ville_departement,
COUNT(ville_commune) AS nbr_items

FROM villes_france_free

JOIN departement ON departement_code =
ville_departement

GROUP BY ville_departement

ORDER BY nbr_items DESC;
```

departement_nom	ville_departement	nbr_items
Pas-de-Calais	62	895
Aisne	02	816
Somme	80	782
Seine-Maritime	76	745
Moselle	57	730
Calvados	14	706
Côte-d'or	21	706
Oise	60	693
Eure	27	675
Nord	59	650
Marne	51	620
Manche	50	601
Doubs	25	594
Meurthe-et-Moselle	54	594

Obtenir la liste des 10 plus grands départements, en terme de superficie

```
SELECT departement_nom, ville_departement,
ROUND(SUM(ville_surface),2) AS sum_surface

FROM villes_france_free

LEFT JOIN departement ON departement_code =
ville_departement

GROUP BY ville_departement

ORDER BY sum_surface DESC

LIMIT 10;
```

departement_nom	ville_departement	sum_surface
Guyane	973	83531
Gironde	33	9975.59
Landes	40	9242.6
Dordogne	24	9060.01
Côte-d'or	21	8763.21
Aveyron	12	8735.12
Saône-et-Loire	71	8574.69
Marne	51	8161.58
Puy-de-Dôme	63	7969.66
Pyrénées-Atlantiques	64	7644.76

Compter le nombre de villes dont le nom commence par "Saint"

```
SELECT ville_nom, COUNT(ville_nom) AS count_ville

FROM villes_france_free

WHERE ville_nom LIKE "SAINT%";
```

ville_nom	count_ville
SAINT BARTHELEMY	4260

Obtenir la liste des villes qui ont un nom existants plusieurs fois, et trier afin d'obtenir en premier celles dont le nom est le plus souvent utilisé par plusieurs communes

3 techniques possibles

```
SELECT ville_nom, COUNT(ville_nom) AS nbt_item
FROM villes_france_free
GROUP BY ville_nom
ORDER BY nbt_item DESC;
```

NB : Affiche par ordre aléatoire le nom des villes ayant la même occurrence

ville_nom	nbt_item
SAINTE-COLOMBE	14
SAINT-SAUVEUR	12
BEAULIEU	11
SAINTE-MARIE	11
SAINT-REMY	10
SAINT-SULPICE	10
SAINT-AUBIN	10
SAINT-LOUP	10
LE PIN	10
SAINT-HILAIRE	9
SAINT-CHRISTOP...	9
SAINT-MICHEL	9
SAINT-PAUL	9
SAINT-MARCEL	9
BEAUMONT	9
SAINT-MEDARD	9
SAINT-HIPPOLYTE	8
SAINT-LEGER	8

Ou

```
SELECT ville_nom, COUNT(ville_nom) AS nombre_occurrences
FROM villes_france_free
GROUP BY ville_nom
HAVING COUNT(ville_nom) > 1
ORDER BY COUNT(ville_nom) DESC;
```

NB : Affiche par ordre alphabétique

ville_nom	nombre_occurrences
SAINTE-COLOMBE	14
SAINT-SAUVEUR	12
BEAULIEU	11
SAINTE-MARIE	11
LE PIN	10
SAINT-AUBIN	10
SAINT-LOUP	10
SAINT-REMY	10
SAINT-SULPICE	10
BEAUMONT	9
SAINT-CHRISTOP...	9
SAINT-HILAIRE	9
SAINT-MARCEL	9
SAINT-MEDARD	9
SAINT-MICHEL	9
SAINT-PAUL	9
BRION	8
SAINT-CLEMENT	8

Ou

```
SELECT ville_nom, COUNT(ville_nom) AS nombre_occurrences
    CASE
        WHEN COUNT(ville_nom) > 1 THEN 'ville_nom apparaît plusieurs fois'
        ELSE 'ville_nom apparaît une seule fois'
    END AS statut
FROM villes_france_free
GROUP BY ville_nom
HAVING COUNT(ville_nom) > 1
ORDER BY COUNT(ville_nom) DESC;
```

ville_nom	nombre_occurrences	statut
SAINTE-COLOMBE	14	ville_nom apparaît plusieurs fois
SAINT-SAUVEUR	12	ville_nom apparaît plusieurs fois
BEAULIEU	11	ville_nom apparaît plusieurs fois
SAINTE-MARIE	11	ville_nom apparaît plusieurs fois
LE PIN	10	ville_nom apparaît plusieurs fois
SAINT-AUBIN	10	ville_nom apparaît plusieurs fois
SAINT-LOUP	10	ville_nom apparaît plusieurs fois
SAINT-REMY	10	ville_nom apparaît plusieurs fois
SAINT-SULPICE	10	ville_nom apparaît plusieurs fois
BEAUMONT	9	ville_nom apparaît plusieurs fois
SAINT-CHRISTOP...	9	ville_nom apparaît plusieurs fois
SAINT-HILAIRE	9	ville_nom apparaît plusieurs fois
SAINT-MARCEL	9	ville_nom apparaît plusieurs fois
SAINT-MEDARD	9	ville_nom apparaît plusieurs fois
SAINT-MICHEL	9	ville_nom apparaît plusieurs fois
SAINT-PAUL	9	ville_nom apparaît plusieurs fois
BRION	8	ville_nom apparaît plusieurs fois
SAINT-CLEMENT	8	ville_nom apparaît plusieurs fois

Obtenir en une seule requête SQL la liste des villes dont la superficie est supérieur à la superficie moyenne

ville_nom	ville_surface
MARIPASOULA	18360
REGINA	12130
CAMOPI	10030
MANA	6332
SAINT-ELIE	5680
SAINT-LAURENT-DU-MARONI	4830
SAUL	4475
ROURA	3902
IRACOUBO	2762
POMPIDOU PAPA ICHTON	2628
SAINT-GEORGES	2320
KOUROU	2160
GRAND-SANTI	2112
APATOU	2020
SINNAMARY	1340
OUANARY	1080
ARLES	758.93
MONTSINERY-TONNEGRANDE	600

```
SELECT ville_nom, ville_surface
FROM villes_france_free
WHERE ville_surface > (SELECT AVG(ville_surface) FROM villes_france_free)
ORDER BY ville_surface DESC;
```

NB : Après un Where si la comparaison n'est pas avec une donnée brute de la table, il faut faire une sous-requête. Même avec MIN, MAX.

Obtenir la liste des départements qui possèdent plus de 2 millions d'habitants

! Ne prendre en compte que l'année la plus récente. Elle correspond au dernier recensement de la population !

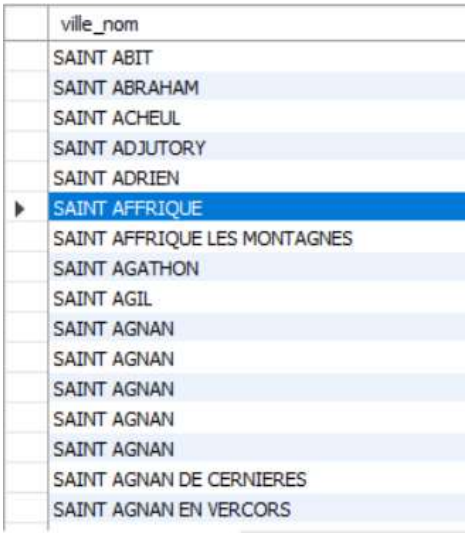
departement_nom	n°_dept	sum_population
Nord	59	2565600
Paris	75	2211000

```
SELECT departement_nom, ville_departement AS n°_dept, SUM(ville_population_2012) AS sum_population
FROM villes_france_free
LEFT JOIN departement ON departement_code = ville_departement
GROUP BY ville_departement
HAVING sum_population > 2000000
ORDER BY sum_population DESC;
```

Remplacez les tirets par un espace vide, pour toutes les villes commençant par “SAINT-” (dans la colonne qui contient les noms en majuscule)

```
UPDATE villes_france_free
SET ville_nom = REPLACE(ville_nom, '-', ' ')
WHERE ville_nom LIKE 'SAINT-%';
```

```
SELECT ville_nom
FROM villes_france_free
WHERE ville_nom LIKE "SAINT%";
```



ville_nom
SAINT ABIT
SAINT ABRAHAM
SAINT ACHEUL
SAINT ADJUTORY
SAINT ADRIEN
SAINT AFFRIQUE
SAINT AFFRIQUE LES MONTAGNES
SAINT AGATHON
SAINT AGIL
SAINT AGNAN
SAINT AGNAN
SAINT AGNAN
SAINT AGNAN
SAINT AGNAN
SAINT AGNAN DE CERNIERES
SAINT AGNAN EN VERCORS

Obtenir l’utilisateur ayant le prénom “Muriel” et le mot de passe “test11”, sachant que l’encodage du mot de passe est effectué avec l’algorithme Sha1.

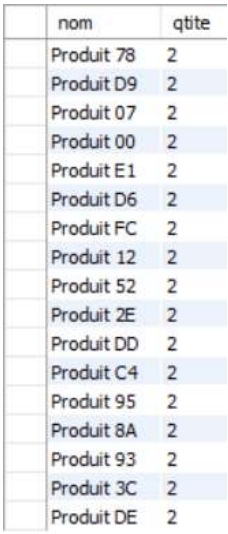
```
SELECT * FROM client
WHERE prenom = 'Muriel' AND password = SHA1('test11');
```

id	prenom	nom	email	ville	password
11	Muriel	Dupuis	muriel@example.com	Paris	100c4e57374fc998e57164d4c0453bd3a4876a58

Obtenir la liste de tous les produits qui sont présent sur plusieurs commandes.

```
SELECT nom, COUNT(quantite) AS qtite
FROM commande_ligne
GROUP BY nom
HAVING qtite > 1
ORDER BY qtite DESC;
```

Le having permet de ne pas compter les cdes présentent une seule fois (respect consigne)



nom	qtite
Produit 78	2
Produit D9	2
Produit 07	2
Produit 00	2
Produit E1	2
Produit D6	2
Produit FC	2
Produit 12	2
Produit 52	2
Produit 2E	2
Produit DD	2
Produit C4	2
Produit 95	2
Produit 8A	2
Produit 93	2
Produit 3C	2
Produit DE	2

Obtenir la liste de tous les produits qui sont présent sur plusieurs commandes et y ajouter une colonne qui liste les identifiants des commandes associées.

```
SELECT nom, COUNT(*) AS qtite , GROUP_CONCAT(commande_id) AS liste_commandes
FROM commande_ligne
GROUP BY nom
HAVING qtite > 1
ORDER BY qtite DESC ;
```

nom	nbr_items	liste_commandes
Produit 6D	4	23,29,40,41
Produit 67	3	15,17,26
Produit 00	2	5,14
Produit 07	2	4,10
Produit 12	2	12,18
Produit 2E	2	16,46
Produit 3C	2	31,36
Produit 52	2	15,42
Produit 78	2	2,4
Produit 8A	2	23,41
Produit 93	2	28,47
Produit 95	2	22,32
Produit C4	2	20,46
Produit D6	2	7,33
Produit D9	2	3,7
Produit DD	2	20,25
Produit DE	2	32,48

Enregistrer le prix total à l'intérieur de chaque ligne des commandes, en fonction du prix unitaire et de la quantité

```
UPDATE commande_ligne
SET prix_total = quantite * prix_unitaire;

SELECT * FROM commande_ligne;
```

id	commande_id	nom	quantite	prix_unitaire	prix_total
1	1	Produit 19	3	49.57	148.71
2	1	Produit 92	4	81.24	324.96
3	1	Produit 68	2	17.48	34.96
4	2	Produit 53	4	83.69	334.76
5	2	Produit 78	6	5.99	35.94
6	3	Produit D9	7	18.91	132.37
7	4	Produit A3	8	76.57	612.56
8	4	Produit BB	10	86.14	861.4
9	4	Produit 7C	4	80.96	323.84
10	4	Produit 78	9	26.4	237.6
11	4	Produit 07	6	9.13	54.78
12	5	Produit 00	10	86.45	864.5
13	5	Produit 7A	2	44.86	89.72
14	6	Produit E1	9	84.93	764.37

Obtenir le montant total pour chaque commande et y voir facilement la date associée à cette commande ainsi que le prénom et nom du client associé

```
SELECT client.prenom, client.nom,
commande.date_achat,
commande_ligne.commande_id,
ROUND(SUM(prix_total),2) AS
prix_commande

FROM commande_ligne

LEFT JOIN commande ON commande.id =
commande_ligne.commande_id

LEFT JOIN client ON client.id =
commande.client_id

GROUP BY commande_id;
```

prenom	nom	date_achat	commande_id	prix_commande
Flavie	Da costa	2019-01-01	1	508.63
Emilien	Camus	2019-01-03	2	370.7
Olivier	Riou	2019-01-04	3	132.37
Lucas	Jung	2019-01-07	4	2090.18
Christiane	Riou	2019-01-08	5	954.22
Amaury	Payet	2019-01-09	6	764.37
Maris	Buisson	2019-01-10	7	1111.64
Emile	Ramos	2019-01-11	8	1000.08
Armel	Vigneron	2019-01-11	9	1129.3
Arnaude	Vallee	2019-01-13	10	1063.17
NULL	NULL	2019-01-14	11	97
NULL	NULL	2019-01-15	12	482.45
NULL	NULL	2019-01-16	13	451.94
NULL	NULL	2019-01-16	14	1223.32
NULL	NULL	2019-01-17	15	1646.31

Enregistrer le montant total de chaque commande dans le champ intitulé “cache_prix_total”

```
UPDATE commande AS t1

INNER JOIN

    ( SELECT commande_id, SUM(commande_ligne.prix_total) AS p_total

      FROM commande_ligne

      GROUP BY commande_id ) t2 ON  t1.id = t2.commande_id

SET t1.cache_prix_total = t2.p_total;


SELECT commande_id, cache_prix_total

FROM commande

LEFT JOIN commande_ligne ON commande.id =
commande_ligne.commande_id

GROUP BY commande_id;
```

commande_id	cache_prix_total
1	508.63
2	370.7
3	132.37
4	2090.18
5	954.22
6	764.37
7	1111.64
8	1000.08
9	1129.3
10	1063.17
11	97
12	482.45
13	451.94
14	1223.32
15	1646.31
16	136.4
17	1285.81

Obtenir le montant global de toutes les commandes, pour chaque mois

```
SELECT MONTH(date_achat), ROUND(SUM(cache_prix_total),2) AS sum_prix_total

FROM commande

GROUP BY  MONTH(date_achat);
```

MONTH(date_achat)	sum_prix_total
1	21259.57
2	18616.68

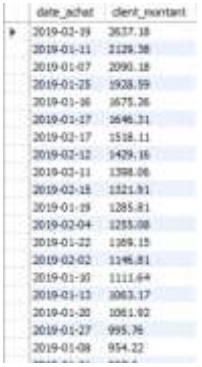
Obtenir la liste des 10 clients qui ont effectué le plus grand montant de commandes, et obtenir ce montant total pour chaque client.

```
SELECT client.nom, client.prenom,
ROUND(SUM(commande.cache_prix_total),2) AS client_montant

FROM `commande`

LEFT JOIN client ON client.id = commande.client_id

GROUP BY commande.client_id

ORDER BY client_montant DESC

LIMIT 10;
```

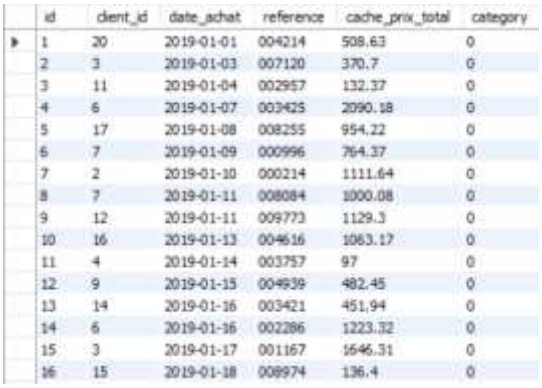
nom	prenom	client_montant
Vespasien	Valentin	5988.18
Saunier	Patrick	3695.36
Collin	Gustave	3625.17
Riou	Olivier	3313.5
Jung	Lucas	3108.41
Riou	Christiane	2886.4
Durand	Manon	2241.08
Huet	Maurice	2182.06
Buisson	Maris	2064.99
Langlois	Jacinthe	1942.59

Obtenir le montant total des commandes pour chaque date

```
SELECT date_achat, ROUND(SUM(cache_prix_total), 2) AS client_montant

FROM commande AS C

GROUP BY C.date_achat

ORDER BY client_montant DESC ;
```



Ajouter une colonne intitulée “category” à la table contenant les commandes. Cette colonne contiendra une valeur numérique

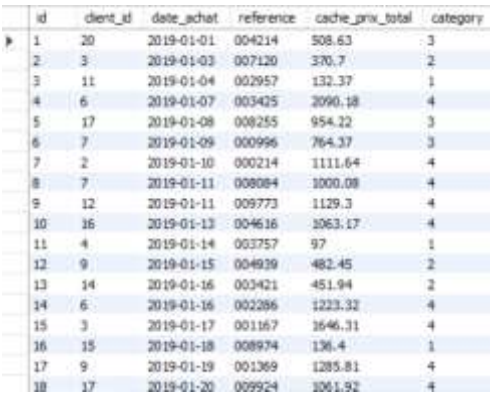


id	client_id	date_achat	reference	cache_prix_total	category
1	20	2019-01-01	004214	508.63	0
2	3	2019-01-03	007120	370.7	0
3	11	2019-01-04	002957	132.37	0
4	6	2019-01-07	003425	2090.18	0
5	17	2019-01-08	008255	954.22	0
6	7	2019-01-09	000996	764.37	0
7	2	2019-01-10	000214	1111.64	0
8	7	2019-01-11	008084	1000.08	0
9	12	2019-01-11	009773	1129.3	0
10	16	2019-01-13	004616	1063.17	0
11	4	2019-01-14	003757	97	0
12	9	2019-01-15	004939	482.45	0
13	14	2019-01-16	003421	451,94	0
14	6	2019-01-16	002286	1223.32	0
15	3	2019-01-17	001167	1646.31	0
16	15	2019-01-18	008974	136.4	0

```
ALTER TABLE commande

ADD category INT NOT NULL;

SELECT * FROM commande;
```

Enregistrer la valeur de la catégorie, en suivant les règles suivantes :

- “1” pour les commandes de moins de 200€
- “2” pour les commandes entre 200€ et 500€
- “3” pour les commandes entre 500€ et 1.000€
- “4” pour les commandes supérieures à 1.000€

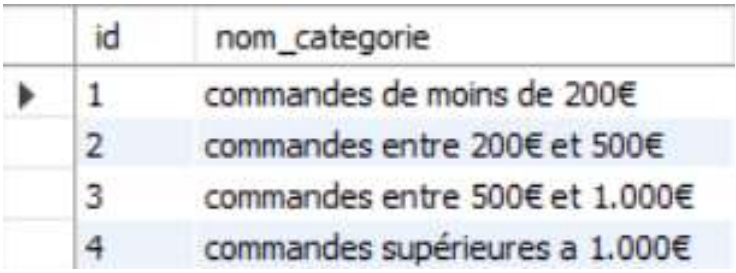


id	client_id	date_achat	reference	cache_prix_total	category
1	20	2019-01-01	004214	508.63	3
2	3	2019-01-03	007120	370.7	2
3	11	2019-01-04	002957	132.37	1
4	6	2019-01-07	003425	2090.18	4
5	17	2019-01-08	008255	954.22	3
6	7	2019-01-09	000996	764.37	3
7	2	2019-01-10	000214	1111.64	4
8	7	2019-01-11	008084	1000.08	4
9	12	2019-01-11	009773	1129.3	4
10	16	2019-01-13	004616	1063.17	4
11	4	2019-01-14	003757	97	1
12	9	2019-01-15	004939	482.45	2
13	14	2019-01-16	003421	451.94	2
14	6	2019-01-16	002286	1223.32	4
15	3	2019-01-17	001167	1646.31	4
16	15	2019-01-18	008974	136.4	1
17	9	2019-01-19	001369	1285.81	4
18	17	2019-01-20	009924	1061.92	4

```
UPDATE commande

Set category =

        CASE

                WHEN cache_prix_total < 200 THEN '1'

      WHEN cache_prix_total BETWEEN 200 AND 500 THEN '2'

      WHEN cache_prix_total BETWEEN 500 AND 1000 THEN '3'

      ELSE '4'

        END ;
```

Créer une table intitulée “commande_category” qui contiendra le descriptif de ces catégories

 # Deux techniques



id	nom_categorie
1	commandes de moins de 200€
2	commandes entre 200€ et 500€
3	commandes entre 500€ et 1.000€
4	commandes supérieures a 1.000€

```
CREATE TABLE commande_category

(

`id` INT NOT NULL AUTO_INCREMENT,

`nom_categorie` VARCHAR(100),

PRIMARY KEY (`id`)

);
```

INSERT INTO commande_category

(`id`, `nom_categorie`)

VALUES (1, "commandes de moins de 200€"),

(2 , "commandes entre 200€ et 500€"),

(3, "commandes entre 500€ et 1.000€"),

(4, "commandes supérieures a 1.000€")

;

Ou

CREATE TABLE `commande_category`

(`id` int(1) NOT NULL AUTO_INCREMENT,

`nom_category` varchar(255) NOT NULL,

PRIMARY KEY (`id`));

INSERT INTO `commande_category` (`id`, `nom_category`) VALUES (1, 'commandes de moins de 200€');

INSERT INTO `commande_category` (`id`, `nom_category`) VALUES (2, 'commandes entre 200€ et 500€');

INSERT INTO `commande_category` (`id`, `nom_category`) VALUES (3, 'commandes entre 500€ et 1.000€');

INSERT INTO `commande_category` (`id`, `nom_category`) VALUES (4, "commandes supérieures a 1.000€") ;

Supprimer toutes les commandes (et les lignes des commandes) inférieur au 1er février 2019. Cela doit être effectué en 2 requêtes maximum

```
DELETE FROM `commande_ligne`
WHERE `commande_id` IN ( SELECT id FROM commande WHERE date_achat < '2019-02-01' );
DELETE FROM `commande` WHERE date_achat < '2019-02-01';
```