

EXOS BD.STORE

1. 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 = encode(digest('test11', 'sha1'), 'hex');
```

```
runpsql - Raccourci
nide » n'existe pas, poursuite du traitement
DROP TABLE
CREATE TABLE
INSERT 0 48
pgsql:C:/Users/saytt/OneDrive/Bureau/WorkSpaceFormationInitial/sqlExo2/commande_psql.sql:62: ATTENTION:  aucune transaction en cours
COMMIT
postgres=# \dt
           Liste des relations
  Schéma |   Nom    | Type | Propriétaire
-----+-----+-----+
public | client   | table | postgres
public | commande | table | postgres
public | commande_ligne | table | postgres
(3 lignes)

postgres=# CREATE EXTENSION IF NOT EXISTS pgcrypto;
CREATE EXTENSION
postgres=# SELECT * FROM client WHERE prenom = 'Muriel' AND password = encode(digest('test11', 'sha1'), 'hex');
 id | prenom | nom      | email        | ville | password
---+-----+-----+-----+-----+-----+
 11 | Muriel | Dupuis  | muriel@example.com | Paris | 100c4e57374fc998e57164d4c0453bd3a4876a58
(1 ligne)

postgres=#

```

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

```
-> select nom, count(*) as produit_presents from commande_ligne group by nom having count(*) > 1
order by produit_presents desc;
```

```
runpsql - Raccourci
postgres=# select nom, count(*) as produit_presents from commande_ligne group by nom having count(*) > 1 order by produit_presents desc;
 nom | produit_presents
-----+-----
Produit 6D |      4
Produit 67 |      3
Produit DD |      2
Produit 52 |      2
Produit DE |      2
Produit D9 |      2
Produit 95 |      2
Produit D6 |      2
Produit 2E |      2
Produit FC |      2
Produit 00 |      2
Produit 3C |      2
Produit E1 |      2
Produit 8A |      2
Produit 12 |      2
Produit 78 |      2
Produit 93 |      2
Produit C4 |      2
Produit 07 |      2
(19 lignes)

postgres=#

```

3. 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(*), array_to_string(array_agg(commande_id  order by commande_id ), ',') from
commande_ligne group by nom having count(*) > 1 order by count(*) desc;
```

```

runpsql - Raccourci
postgres=# select nom, count(*), array_to_string(array_agg(commande_id order by commande_id ), ',') from
  commande_ligne group by nom having count(*) > 1 order by count(*) desc;
   nom      | count | array_to_string
-----+-----+-----
Produit 6D |     4 | 23,29,40,41
Produit 67 |     3 | 15,17,26
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
Produit E1 |     2 | 6,22
Produit 00 |     2 | 5,14
Produit FC |     2 | 10,21
Produit 07 |     2 | 4,10
(19 lignes)

postgres=

```

4. 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);

```

runpsql - Raccourci
postgres=# update commande_ligne set prix_total = (quantite * prix_unitaire);
UPDATE 120
postgres=# select prix_unitaire from commande_ligne;
   prix_unitaire
-----
 49.57
 81.24
 17.48
 83.69
 5.99
 18.91
 76.57
 86.14
 80.96
 26.4
 9.13
 86.45
 44.86
 84.93
 50.07
 115.55
 67.55
 111.12
 112.93
 111.31
 97.75

```

5. 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, SUM(prix_total) AS total_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 client.nom, client.prenom,
 commande.date_achat ORDER BY nom;

runpsql - Raccourci

```
nde_ligne LEFT JOIN commande ON commande.id = commande_ligne.commande_id LEFT JOIN client ON client.id = co
mmande.client_id GROUP BY client.nom, client.prenom, commande.date_achat ORDER BY nom;
```

prenom	nom	date_achat	total_commande
Maris	Buisson	2019-01-18	136.4
Maris	Buisson	2019-01-25	1928.59
Emilien	Camus	2019-01-14	97
Emilien	Camus	2019-01-27	995.76
Emilien	Camus	2019-02-13	719.5400000000001
Gustave	Collin	2019-01-03	370.7
Gustave	Collin	2019-01-17	1646.3100000000002
Gustave	Collin	2019-01-21	907.2
Gustave	Collin	2019-02-06	700.96
Muriel	Dupuis	2019-01-04	132.37
Muriel	Dupuis	2019-02-02	362.81
Muriel	Dupuis	2019-02-03	673.65
Manon	Durand	2019-01-15	482.4500000000005
Manon	Durand	2019-01-19	1285.8100000000002
Manon	Durand	2019-02-01	472.82
Fabrice	Foucher	2019-01-13	1063.17
Fabrice	Foucher	2019-02-09	554.7
Maurice	Huet	2019-02-02	784
Maurice	Huet	2019-02-11	1398.06
Lucas	Jung	2019-01-09	764.3700000000001
Lucas	Jung	2019-01-11	1000.08
Lucas	Jung	2019-02-05	751.6400000000001
Lucas	Jung	2019-02-16	592.3199999999999
Jacinthe	Langlois	2019-02-14	620.6800000000001
Jacinthe	Langlois	2019-02-15	1321.9099999999999

6. (difficulté très haute) Enregistrer le montant total de chaque commande dans le champ intitulé "cache_prix_total"

```
-> update commande set cache_prix_total = t2.p_total from (select commande_id,
sum(commande_ligne.prix_total) as p_total from commande_ligne group by commande_id) as t2 where
commande.id = t2.commande_id;
```

runpsql - Raccourci

```
store=# update commande set cache_prix_total = t2.p_total from (select commande_id, sum(commande_ligne.prix_total)
l) as p_total from commande_ligne group by commande_id) as t2 where commande.id = t2.commande_id;
UPDATE 48
store=# select * from commande;
ERREUR: erreur de syntaxe sur ou près de « commande »
LINE 1 : select * from commande;
^

store=# select * from commande;
id | client_id | date_achat | reference | cache_prix_total
-----+-----+-----+-----+-----+
1 | 20 | 2019-01-01 | 004214 | 508.6299999999994
2 | 3 | 2019-01-03 | 007120 | 370.7
3 | 11 | 2019-01-04 | 002957 | 132.37
4 | 6 | 2019-01-07 | 003425 | 2090.18
5 | 17 | 2019-01-08 | 008255 | 954.22
6 | 7 | 2019-01-09 | 000996 | 764.3700000000001
7 | 2 | 2019-01-10 | 000214 | 1111.639999999999
8 | 7 | 2019-01-11 | 008084 | 1000.08
9 | 12 | 2019-01-11 | 009773 | 1129.3000000000002
10 | 16 | 2019-01-13 | 004616 | 1063.1699999999998
```

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

```
-> select extract(MONTH from date_achat) as mois, sum(cache_prix_total) from commande group by
mois;
```

```
runpsql - Raccourci

store=# select extract(MONTH from date_achat), sum(cache_prix_total) from commande group by MONTH order by asc;
ERREUR: erreur de syntaxe sur ou près de « asc »
LIGNE 1 : ...cache_prix_total) from commande group by MONTH order by asc;

store=# select extract(MONTH from date_achat), sum(cache_prix_total) from commande group by MONTH;
ERREUR: la colonne « month » n'existe pas
LIGNE 1 : ..._achat), sum(cache_prix_total) from commande group by MONTH;

store=# select extract(MONTH from date_achat) as mois, sum(cache_prix_total) from commande group by mois;
mois | sum
-----+-----
 1 | 21259.57
 2 | 18616.68
(2 lignes)

store=#

```

8. 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, sum(commande.cache_prix_total) as client_montant from commande left join client on client.id = commande.client_id group by client.nom, client.prenom, commande.client_id order by client_montant desc limit 10;

```
runpsql - Raccourci

store=# select client.nom, client.prenom, sum(commande.cache_prix_total) as client_montant from commande left join client on client.id = commande.client_id group by client.nom, client.prenom, commande.client_id order by client_montant desc limit 10;
nom | prenom | client_montant
-----+-----+-----
Vespasien | Valentin | 5988.179999999999
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
(10 lignes)

store=#

```

9. Obtenir le montant total des commandes pour chaque date

-> select date_achat, sum(cache_prix_total) as montant_date from commande group by date_achat order by date_achat asc;

```

store=# select date_achat, sum(cache_prix_total) from commande group by date_achat;
+-----+-----+
| date_achat | sum |
+-----+-----+
| 2019-02-16 | 592.3199999999999 |
| 2019-02-08 | 903.8799999999999 |
| 2019-02-03 | 673.65 |
| 2019-02-07 | 441.85 |
| 2019-01-21 | 907.2 |
| 2019-01-16 | 1675.26 |
| 2019-01-18 | 136.4 |
| 2019-01-10 | 1111.6399999999999 |
| 2019-02-17 | 1518.11 |
| 2019-01-01 | 508.6299999999994 |
| 2019-01-20 | 1061.9199999999998 |
| 2019-01-25 | 1928.59 |
| 2019-02-18 | 611.52 |
| 2019-02-15 | 1321.9099999999999 |
| 2019-01-17 | 1646.3100000000002 |
| 2019-01-14 | 97 |
+-----+-----+

```

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

-> alter table commande add column category INT;

```

store=# alter table commande add column category INT;
ALTER TABLE
store=# select * from commande;
+----+----+----+----+----+----+
| id | client_id | date_achat | reference | cache_prix_total | category |
+----+----+----+----+----+----+
| 1 | 20 | 2019-01-01 | 004214 | 508.6299999999994 |          |
| 2 | 3 | 2019-01-03 | 007120 | 370.7 |          |
| 3 | 11 | 2019-01-04 | 002957 | 132.37 |          |
| 4 | 6 | 2019-01-07 | 003425 | 2090.18 |          |
| 5 | 17 | 2019-01-08 | 008255 | 954.22 |          |
| 6 | 7 | 2019-01-09 | 000996 | 764.3700000000001 |          |
| 7 | 2 | 2019-01-10 | 000214 | 1111.6399999999999 |          |
| 8 | 7 | 2019-01-11 | 008084 | 1000.08 |          |
| 9 | 12 | 2019-01-11 | 009773 | 1129.3000000000002 |          |
| 10 | 16 | 2019-01-13 | 004616 | 1063.1699999999998 |          |
| 11 | 4 | 2019-01-14 | 003757 | 97 |          |
| 12 | 9 | 2019-01-15 | 004939 | 482.4500000000005 |          |
| 13 | 14 | 2019-01-16 | 003421 | 451.94 |          |
| 14 | 6 | 2019-01-16 | 002286 | 1223.32 |          |
+----+----+----+----+----+----+

```

11. 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€

-> update commande set category = (case when cache_prix_total<200 then 1 when cache_prix_total<500 then 2 when cache_prix_total<1000 then 3 else 4 end);

17	9	2019-01-19	001369	1285.81	
^C	store=# update commande set category = (case when cache_prix_total<200 then 1 when cache_prix_tot				
al<500 then 2 when cache_prix_total<1000 then 3 else 4 end);					
UPDATE 48					
store=# select * from commande;					
id client_id date_achat reference cache_prix_total category					
-----	-----	-----	-----	-----	-----
1 20 2019-01-01 004214 508.6299999999994 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 764.3700000000001 3					
7 2 2019-01-10 000214 1111.6399999999999 4					
8 7 2019-01-11 008084 1000.08 4					
9 12 2019-01-11 009773 1129.3000000000002 4					
10 16 2019-01-13 004616 1063.1699999999998 4					
11 4 2019-01-14 003757 97 1					
12 9 2019-01-15 004939 482.4500000000005 2					
13 14 2019-01-16 003421 451.94 2					

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

-> create table commande_category(id serial primary key, nom varchar(255) not null);

LIGNE 1 : create table commande_category(id int not null auto_incremen...
^C
store=# create table commande_category(id int not null serial, nom varchar(255) not null, primary key(id));
ERREUR: erreur de syntaxe sur ou près de « serial »
LIGNE 1 : create table commande_category(id int not null serial, nom v...
^C
store=# create table commande_category(id int not null serial primary key, nom varchar(255) not null);
ERREUR: erreur de syntaxe sur ou près de « serial »
LIGNE 1 : create table commande_category(id int not null serial primar...
^C
store=# create table commande_category(id serial primary key, nom varchar(255) not null);
CREATE TABLE
store=# select * from commande_category;
id nom
(0 ligne)

13. Insérer les 4 descriptifs de chaque catégorie au sein de la table précédemment créée

-> insert into commande_category (id, nom) values(1, 'commandes de moins de 200€');

-> insert into commande_category (id, nom) values(2, 'commandes entre 200€ et 500€');

-> insert into commande_category (id, nom) values(3, 'commandes entre 500€ et 1000€');

-> insert into commande_category (id, nom) values(4, 'commandes supérieur à 1000€');

```
runpsql - Raccourci
store=# insert into commande_category (id, nom) values(1, 'commandes de moins de 200€');
INSERT 0 1
store=# insert into commande_category (id, nom) values(2, 'commandes entre 200€ et 500€');
INSERT 0 1
store=# insert into commande_category (id, nom) values(3, 'commandes entre 500€ et 1000€');
INSERT 0 1
store=# insert into commande_category (id, nom) values(4, 'commandes supérieur à 1000€');
INSERT 0 1
store=# select * from commande_category;
 id |          nom
----+
 1 | commandes de moins de 200€
 2 | commandes entre 200€ et 500€
 3 | commandes entre 500€ et 1000€
 4 | commandes supérieur à 1000€
(4 lignes)

store=#

```

14. 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';

```
runpsql - Raccourci
store=# delete from commande_ligne where commande_id in (select id from commande where date_achat < '2019-02-01');
DELETE 58
store=# delete from commande where date_achat < '2019-02-01';
DELETE 24
store=# select * from commande;
 id | client_id | date_achat | reference | cache_prix_total | category
----+
 25 |      9 | 2019-02-01 | 007879 |           472.82 | 2
 26 |      8 | 2019-02-02 | 007277 |            784 | 3
 27 |     11 | 2019-02-02 | 002745 |           362.81 | 2
 28 |     11 | 2019-02-03 | 001893 |           673.65 | 3
 29 |     20 | 2019-02-04 | 001230 |          1255.08 | 4
 30 |     10 | 2019-02-05 | 000469 |            114.4 | 1
 31 |      7 | 2019-02-05 | 008653 | 751.6400000000001 | 3
 32 |      3 | 2019-02-06 | 001858 | 700.9599999999999 | 3
 33 |     14 | 2019-02-07 | 003330 |           441.85 | 2
 34 |      2 | 2019-02-08 | 001074 | 810.1999999999999 | 3
 35 |      5 | 2019-02-08 | 005379 |           93.68 | 1
 36 |     16 | 2019-02-09 | 003672 |           554.7 | 3

```