

## Environnement

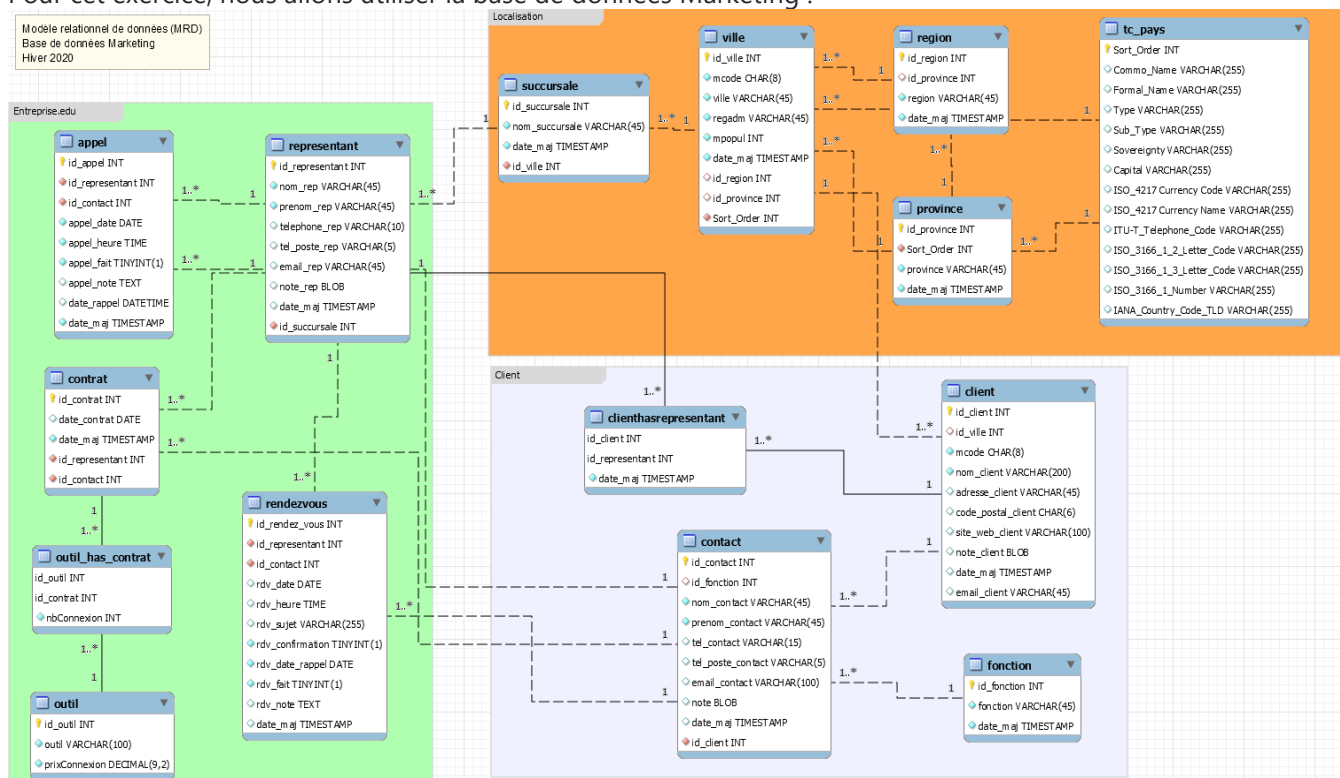
- MySQL Server
- MySQL Workbench

## Directive

- Sauvegardez vos réponses texte et vos instructions SQL dans des fichiers aux noms module05\_ex1.sql

## Exercice 1 - Base de données Marketing

Pour cet exercice, nous allons utiliser la base de données Marketing :



- Connectez vous à votre serveur MySQL
- Exécutez le fichier SQL [Marketing.sql](#)
- Essayez quelques requêtes de base pour bien connaître la bd :

```
USE marketing;

SHOW TABLES;
SELECT * FROM representant;
SELECT COUNT(*) FROM representant;
/* Attention : la fonction COUNT() demande un paramètre soit une champ, en donnant
le nom du champs ou tous les champs (*). */

SELECT * FROM succursale;
SELECT COUNT(*) FROM succursale;
```

- Qu'elle requête me permet de savoir dans quelle succursale sont les représentants ? Essayons comme ceci (avec des ajouts dans la requête) :

```
SELECT
    CONCAT(nom_rep, ' ', prenom_rep) AS Nom, -- Fonction de concaténation.
    id_succursale
FROM
    representant
ORDER BY
    CONCAT(nom_rep, ' ', prenom_rep);
-- On doit faire le tri sur le l'ensemble et non sur l'alias.
```

Cette requête nous donne la clef étrangère `id_succursale`, mais pas le nom de la succursale. Il est nécessaire de créer une jointure avec la table `succursale` pour avoir le nom de la succursale.

## Exercice 1 - Principe des jointures

Le principe des jointures est de joindre plusieurs tables. Pour ce faire, on utilise les informations communes des tables, soit les clefs primaires et les clefs étrangères.

En fait, lorsqu'on fait une jointure, on crée une table virtuelle et temporaire qui reprend les colonnes des tables liées.

```
SELECT *
FROM succursale
    INNER JOIN representant
    ON succursale.id_succursale = representant.id_succursale;
```

Voici ce que vous devriez voir :

	id_succursale	nom_succursale	date_maj	id_ville	id_representant	nom_rep	prenom_rep	telephone_rep	tel_poste_rep	email_rep	note_rep	date_maj	id_succursale
►	1	Québec	2019-02-09 17:38:07	245	1	Duchesneau	Jean-Pierre	4444444444	p303	jpduchesneau@etudiant.édu	BLOB	2009-07-14 18:17:23	1
	1	Québec	2019-02-09 17:38:07	245	2	Desjardins	Alphonse	4186666666	11111	adesjardins@etudiant.édu	BLOB	2009-07-14 18:26:42	1
	2	Montréal	2019-02-09 17:38:07	732	3	Badenas	Louis	5143333333	33333	lbadenas@etudiant.édu	BLOB	2009-07-14 18:33:15	2
	4	Saguenay	2019-02-09 17:38:07	1057	4	Tremblay	François	4184444444	55555	ftremblay@etudiant.édu	BLOB	2009-07-14 18:34:40	4
	3	Rimouski	2019-02-09 17:38:07	95	5	Roy	Guy	NULL	NULL	NULL	NULL	2009-07-14 18:37:30	3
	7	Gaspé	2019-02-09 17:38:07	8	6	De Gaspé	Jean	4185555555	NULL	NULL	NULL	2009-07-14 18:38:41	7
	6	Sherbrooke	2019-02-09 17:38:07	480	7	Daniel	Maire	NULL	NULL	NULL	NULL	2009-07-14 18:40:04	6
	5	Trois-Rivières	2019-02-09 17:38:07	396	8	Roy	Guy	4444444444	509	NULL	NULL	2013-10-25 13:19:53	5
	1	Québec	2019-02-09 17:38:07	245	9	Ricot	Alain	4186666666	NULL	NULL	NULL	2013-10-25 13:20:41	1
	2	Montréal	2019-02-09 17:38:07	732	10	Beaudoin	Jacques	NULL	NULL	NULL	NULL	2013-11-11 21:55:26	2
	4	Saguenay	2019-02-09 17:38:07	1057	11	Louise	Tremblay	NULL	NULL	NULL	NULL	2013-11-11 21:55:58	4
	2	Montréal	2019-02-09 17:38:07	732	12	Duhamel	Louis	NULL	NULL	NULL	NULL	2013-11-11 21:56:11	2

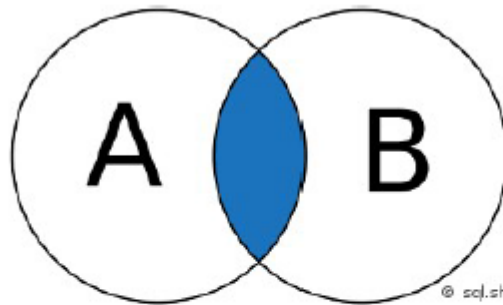
Remarquez que dans le résultat, les quatre champs de la table `succursale` sont "ajoutés" à tous les champs de la table `representant`. Vous retrouvez deux champs `id_succursale`. Pourquoi ?

Le premier est le champ provenant de la table `succursale` et le second celui de la table `representant` (la clef étrangère).

## Exercice 1 - Jointure interne

La clause `INNER JOIN` permet donc de faire une **jointure interne** sur deux tables.

Lorsqu'on fait une jointure interne, cela veut dire qu'on exige qu'il y ait des données de part et d'autre de la jointure; donc, si l'on fait une jointure sur la colonne `a` de la table `A` et la colonne `b` de la table `B` :



```
SELECT *  
FROM A  
INNER JOIN B ON A.key = B.key
```

### Notion d'alias

Les alias sont des noms de remplacement, que l'on donne de manière temporaire (le temps d'une requête en fait) à une colonne, une table ou une donnée. Les alias sont introduits par le mot-clef `AS`.

Essayons à nouveau notre requête avec des alias dans les jointures :

```
SELECT *  
FROM succursale AS suc  
    INNER JOIN representant AS rep  
    ON suc.id_succursale = rep.id_succursale;
```

Il s'agit exactement de la même requête avec une simplification des noms de tables. Comme précisé plus haut, les alias peuvent aussi se placer sur les noms de champs et les données.

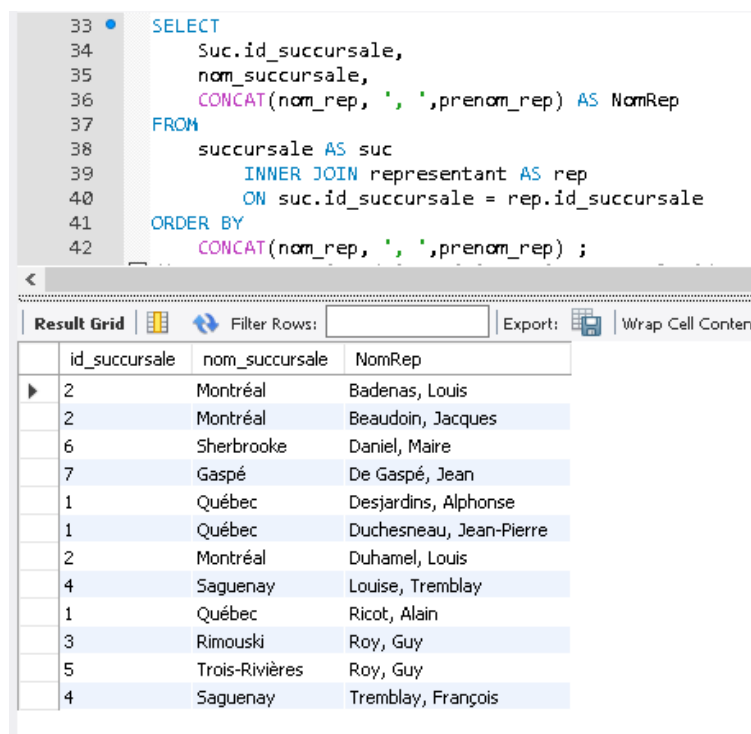
On doit éviter d'utiliser les requêtes qui renvoient l'ensemble des champs. Il vaut mieux préciser ceux dont nous avons besoin. Par exemple :

```

SELECT
    Suc.id_succursale,
    nom_succursale,
    CONCAT(nom_rep, ' ', prenom_rep) AS NomRep
FROM
    succursale AS suc
    INNER JOIN representant AS rep
        ON suc.id_succursale = rep.id_succursale
ORDER BY
    CONCAT(nom_rep, ' ', prenom_rep);
/* Remarquez qu'on doit préciser si on veut le id_succursale de la table succursale
ou celui de la table représentant. Sinon nous aurons l'erreur suivante : Nom de
colonne 'id_succursale' ambigu.

```

On ajoute une clause ORDER BY pour avoir la sortie en ordre de représentant.\*/  
 Résultat :



```

33 SELECT
34     Suc.id_succursale,
35     nom_succursale,
36     CONCAT(nom_rep, ' ', prenom_rep) AS NomRep
37 FROM
38     succursale AS suc
39     INNER JOIN representant AS rep
40         ON suc.id_succursale = rep.id_succursale
41 ORDER BY
42     CONCAT(nom_rep, ' ', prenom_rep) ;

```

id_succursale	nom_succursale	NomRep
2	Montréal	Badenas, Louis
2	Montréal	Beaudoin, Jacques
6	Sherbrooke	Daniel, Maire
7	Gaspé	De Gaspé, Jean
1	Québec	Desjardins, Alphonse
1	Québec	Duchesneau, Jean-Pierre
2	Montréal	Duhamel, Louis
4	Saguenay	Louise, Tremblay
1	Québec	Ricot, Alain
3	Rimouski	Roy, Guy
5	Trois-Rivières	Roy, Guy
4	Saguenay	Tremblay, François

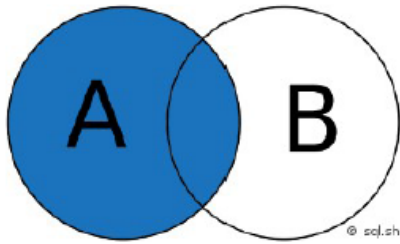
- Trouvez le code pour afficher la même requête en ordre croissant de succursale et en ordre décroissant de nom de représentant en deuxième clef.

## Exercice 1 - Jointure externe

Essayons maintenant de répondre à la question suivante :

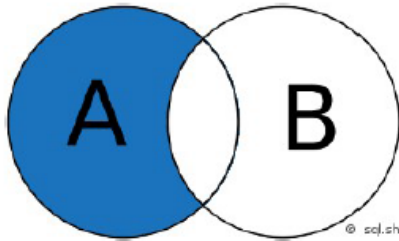
Quelles sont les succursales qui n'ont pas de représentant ?

Pour bien comprendre, regardons ce dessin :



```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
```

LEFT JOIN (sans l'intersection de B)



```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
WHERE B.key IS NUL
```

Donc ce que nous voulons, c'est la jointure gauche, sans les id\_representant, le dessins du bas :

```
SELECT
    nom_succursale
FROM succursale AS suc
    LEFT JOIN representant AS rep
        ON suc.id_succursale = rep.id_succursale
WHERE
    id_representant is null;
```

Vous ne devriez avoir qu'un seul enregistrement, "Sept-Iles", car toutes les autres succursales ont un représentant.

Ajoutons un représentant à la succursale de "Sept-Iles" pour vérifier le bon fonctionnement de notre jointure :

```
INSERT INTO representant
    (nom_rep,prenom_rep,id_succursale)
VALUES
    ('Terry', 'Waspistan', 8);
```

Nous insérons seulement les champs obligatoire (NOT NULL).

Nous laissons les valeurs par défauts agir dans les cas suivants :

- id\_representant : il est auto incrémenté
- date\_maj : valeur par défaut CURRENT\_TIMESTAMP

Pour vous convaincre du bon fonctionnement de ces valeurs, essayez un SELECT après l'insertion.

```
SELECT * FROM representant ;
-- Remarquer l'id_représentant et la date_maj.
```

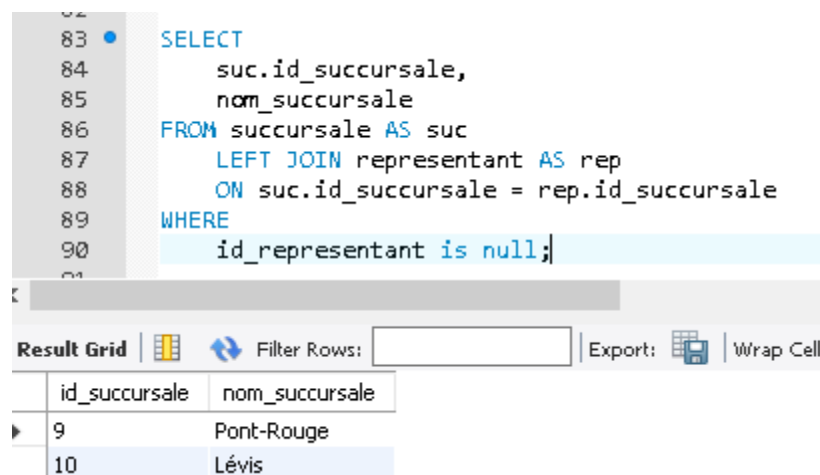
Maintenant, toujours pour vérifier le bon fonctionnement de notre requête de départ (LEFT JOIN) insérons de nouvelles succursales en exécutant cette requête :

```
INSERT INTO succursale (nom_succursale,id_ville)
VALUE
('Pont-Rouge', (SELECT id_ville FROM ville WHERE ville LIKE 'Pont-Rouge')),
('Lévis', (SELECT id_ville FROM ville WHERE ville LIKE 'Lévis'));
```

**Attention :** ici nous avons utilisé des requêtes SELECT pour trouver les deux id\_ville correspondants à nos succursales. Cette façon de faire est très courante dans les requêtes d'insertion.

```
SELECT
    suc.id_succursale,
    nom_succursale
FROM succursale AS suc
    LEFT JOIN representant AS rep
        ON suc.id_succursale = rep.id_succursale
WHERE
    id_representant IS NULL;
```

Voici maintenant ce que vous devriez avoir :



```
83 SELECT
84     suc.id_succursale,
85     nom_succursale
86 FROM succursale AS suc
87     LEFT JOIN representant AS rep
88         ON suc.id_succursale = rep.id_succursale
89 WHERE
90     id_representant is null;
```

	id_succursale	nom_succursale
9	9	Pont-Rouge
10	10	Lévis

Pour avoir une jointure externe vers la droite, il faudrait permettre qu'un représentant puisse exister sans être dans une succursale.

Est-ce le cas ?

```
SELECT
    concat (nom_rep, ' ', prenom_rep) AS NomRep
FROM succursale AS suc
    RIGHT JOIN representant AS rep
        ON suc.id_succursale = rep.id_succursale
WHERE rep.id_succursale IS NULL;
```

Nous allons insérer un représentant sans succursale.

Mais si nous visualisons le schéma de la base de données marketing, nous allons remarquer que notre clef étrangère id\_susscursale est NOT NULL, donc obligatoire.

Nous allons la modifier pour le besoin de la présentation avec le code suivante :

```
ALTER TABLE representant MODIFY COLUMN id_succursale INT NULL;
```

Il est maintenant permis d'entrer un représentant sans lui attribuer une succursale.

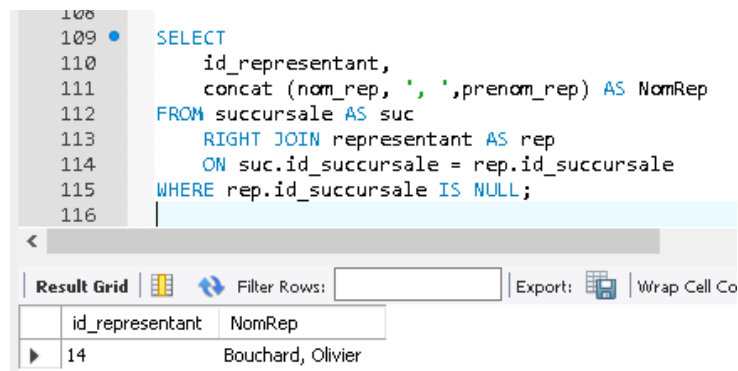
Allons-y avec ce code :

```
INSERT INTO representant
(nom_rep, prenom_rep, telephone_rep, tel_poste_rep, email_rep, note_rep, date_maj,
id_succursale)
VALUES('Bouchard', 'Olivier', NULL, NULL, 'obouchard@gmail.com', NULL, NULL, NULL);
```

Vérifions à nouveau notre requête nous permettant de savoir si un représentant existe sans succursale :

```
SELECT
    id_representant,
    concat (nom_rep, ', ', prenom_rep) AS NomRep
FROM succursale AS suc
    RIGHT JOIN representant AS rep
        ON suc.id_succursale = rep.id_succursale
WHERE rep.id_succursale IS NULL;
```

Cette fois nous avons cette réponse :



The screenshot shows a SQL query editor with a query window and a results grid. The query is the same as the one in the previous block. The results grid shows one row of data.

id_representant	NomRep
14	Bouchard, Olivier

## Exercice 1 - Jointures internes avec plus de deux tables

Il est possible de compliquer davantage en ajoutant une troisième table à la requête.

Nous voulons maintenant savoir ceux qui ont des contrats signés :

```
SELECT
    suc.nom_succursale AS 'Nom succursale',
    concat (nom_rep, ', ', prenom_rep) AS 'Nom représentant',
    id_contrat AS 'No. contrat',
    CAST(date_contrat AS DATE) AS 'Date' -- fonction cast, vous connaissez !
FROM representant AS Rep
    INNER JOIN contrat AS Ct
        ON Rep.id_representant = Ct.id_representant
    INNER JOIN succursale AS Suc
        ON Rep.id_succursale=Suc.id_succursale;
```

Cette fois nous avons cette réponse :

```
119 • SELECT suc.nom_succursale as 'Nom succursale',
120       concat (nom_rep, ', ', prenom_rep) as 'Nom représentant',
121       id_contrat as 'No. contrat',
122       cast(date_contrat as Date) as 'Date'
123 FROM representant as Rep
124       inner join contrat as Ct
125       on Rep.id_representant = Ct.id_representant
126       inner join succursale as Suc
127       on Rep.id_succursale=Suc.id_succursale;
128
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Nom succursale	Nom représentant	No. contrat	Date
▶	Québec	Duchesneau, Jean-Pierre	27	2018-07-30
	Québec	Duchesneau, Jean-Pierre	30	2018-10-16
	Québec	Duchesneau, Jean-Pierre	41	2019-01-12
	Québec	Duchesneau, Jean-Pierre	56	2018-02-08
	Québec	Duchesneau, Jean-Pierre	60	2018-12-30
	Québec	Duchesneau, Jean-Pierre	80	2018-03-22
	Québec	Duchesneau, Jean-Pierre	84	2018-07-06
	Québec	Duchesneau, Jean-Pierre	97	2018-12-18
	Québec	Duchesneau, Jean-Pierre	110	2018-04-30
	Québec	Duchesneau, Jean-Pierre	116	2018-06-24

Avec 400 enregistrements retournés (400 row(s) returned)

## Exercice 1 - Découverte préparatoire - GROUP BY

Si on veut donner la somme des contrats par succursale, on va donc devoir ajouter deux nouvelles tables, la table détaillée des contrats (outil\_has\_contrat) et outil.

La requête suivante fait appel aux 5 (cinq) tables ainsi qu'à la clause GROUP BY.

La clause GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de calcul sur un groupe de résultat. Ici à la somme par succursale.

```
SELECT
    suc.nom_succursale AS 'Nom succursale',
    SUM(det.nbConnexion * ot.prixConnexion) AS 'Montant des contrats'
    /* La somme des contrat par succursale. On doit pour ce faire allez calculé
chaque connexion.*/
FROM
    outil AS ot
    INNER JOIN outil_has_contrat AS det
        ON ot.id_outil = det.id_outil
    INNER JOIN contrat ct
        ON det.id_contrat = ct.id_contrat
    INNER JOIN representant AS rep
        ON rep.id_representant = ct.id_representant
    INNER JOIN succursale AS suc
        ON rep.id_succursale = suc.id_succursale
GROUP BY suc.nom_succursale WITH ROLLUP; -- Groupé par succursale avec rollup
(cumul)
```



Voici le résultat :

```
130
131 • SELECT
132     Suc.nom_succursale AS 'Nom succursale',
133     SUM(Det.nbConnexion * Ot.prixConnexion) AS 'Montant des contrats'
134 FROM outil as ot
135     inner join outil_has_contrat as Det
136     on ot.id_outil = Det.id_outil
137     inner join contrat Ct
138     on Det.id_contrat = Ct.id_contrat
139     inner join representant as Rep
140     on Rep.id_representant = Ct.id_representant
141     inner join succursale AS Suc
142     on Rep.id_succursale = Suc.id_succursale
143 GROUP BY Suc.nom_succursale with rollup;
144
```

< Result Grid Filter Rows: Export: Wrap Cell Content:

	Nom succursale	Montant des contrats
►	Gaspé	14992.23
	Montréal	31720.24
	Québec	31352.20
	Rimouski	13222.44
	Saguenay	24993.86
	Sherbrooke	11866.11
	Trois-Rivières	15875.94
	NULL	144023.02

## Utilisation d'autres fonctions de statistiques

Il existe plusieurs fonctions qui peuvent être utilisées pour manipuler plusieurs enregistrements, il s'agit des fonctions d'agrégations statistiques, les principales sont les suivantes :

- **AVG()** pour calculer la moyenne d'un set de valeur. Elle permet de connaître le prix du panier moyen pour chaque client.
- **COUNT()** pour compter le nombre de lignes concernées. Elle permet de savoir combien d'achats ont été effectués par chaque client.
- **MAX()** pour récupérer la plus haute valeur. Elle est pratique pour savoir l'achat le plus cher.
- **MIN()** pour récupérer la plus petite valeur. Elle est utile par exemple pour connaître la date du premier achat d'un client.
- **SUM()** pour calculer la somme de plusieurs lignes. Elle permet par exemple de connaître le total de tous les achats d'un client.

Ces petites fonctions se révèlent rapidement indispensables pour travailler sur des données.

## Exercice 1 - Découverte préparatoire - HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

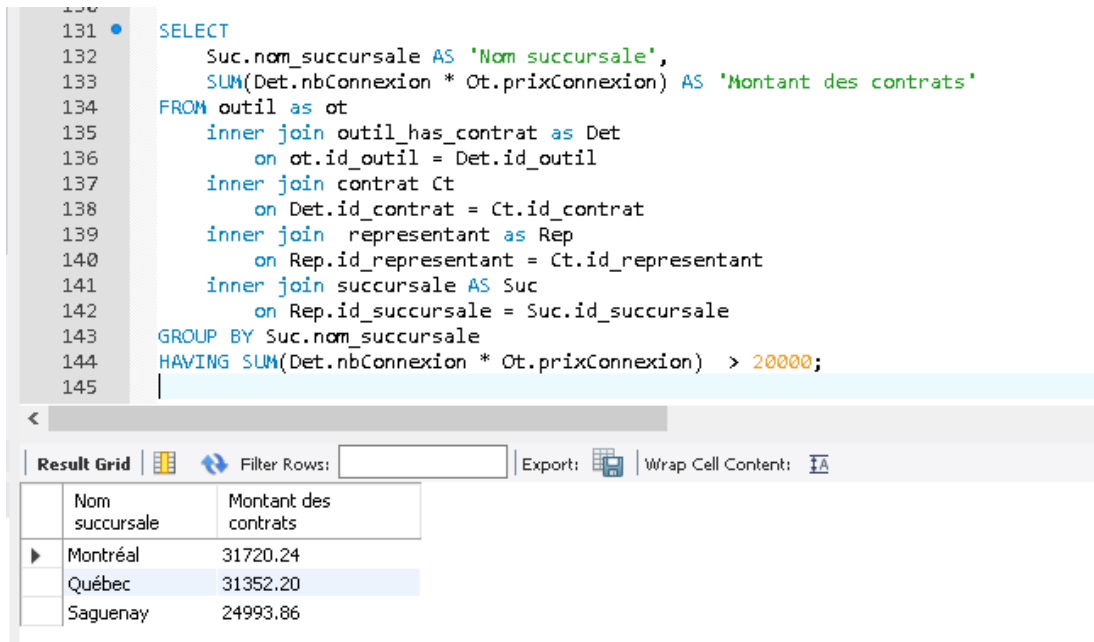
Exemple d'utilisation :

```

SELECT
    suc.nom_succursale AS 'Nom succursale',
    SUM(det.nbConnexion * ot.prixConnexion) AS 'Montant des contrats'
FROM outil AS ot
    INNER JOIN outil_has_contrat AS det
        ON ot.id_outil = det.id_outil
    INNER JOIN contrat ct
        ON det.id_contrat = ct.id_contrat
    INNER JOIN representant AS Rep
        ON Rep.id_representant = ct.id_representant
    INNER JOIN succursale AS suc
        ON Rep.id_succursale = suc.id_succursale
GROUP BY suc.nom_succursale
HAVING SUM(det.nbConnexion * ot.prixConnexion) > 20000;

```

Voici le résultat :



The screenshot shows a SQL query editor with the following query:

```

SELECT
    Suc.nom_succursale AS 'Nom succursale',
    SUM(Det.nbConnexion * Ot.prixConnexion) AS 'Montant des contrats'
FROM outil as ot
    inner join outil_has_contrat as Det
        on ot.id_outil = Det.id_outil
    inner join contrat Ct
        on Det.id_contrat = Ct.id_contrat
    inner join representant as Rep
        on Rep.id_representant = Ct.id_representant
    inner join succursale AS Suc
        on Rep.id_succursale = Suc.id_succursale
GROUP BY Suc.nom_succursale
HAVING SUM(Det.nbConnexion * Ot.prixConnexion) > 20000;

```

Below the query, a table titled 'Result Grid' displays the results:

Nom succursale	Montant des contrats
Montréal	31720.24
Québec	31352.20
Saguenay	24993.86

## Exercice 1 - À vous !

Afficher la liste des clients dans les trois (3) formats suivants en incluant les champs nom\_client, adresse\_client, ville, province, pays, code\_postale :

1. Par nom de client
2. Par code postal
3. Par région administrative du Québec

## Exercice 1 - À vous ! Requêtes complexes

1. Affichez les numéros de succursale, le nom de la succursale, et le nom du représentant concaténé (nom, prénom) dans l'ordre du nom succursale ascendant et comme deuxième ordre de tri nom représentant croissant et ce, en respectant les libellés : **No. succursale, Succursale, Nom représentant.**
2. Affichez la somme des contrats pour chaque client en utilisant les informations suivantes dans l'ordre et en respectant les libellés (clause Alias): **Nom du client, Montant total des contrats.**
  - A-t-on la preuve que les résultats renvoyés sont bons ? **Non.**
  - Vérifions à partir d'une autre requête :
    - Donnez le montant de chacun des contrats pour le "CDE Collège". Additionnez ses montants et vérifiez avec la requête précédente pour voir si les résultats du GROUP BY sont bons.
3. Affichez la somme des contrats en ordre décroissant pour chaque représentant en utilisant les informations suivantes dans l'ordre et en respectant les libellés : **Nom représentant, succursale, Total des contrats.**

### Personnes ayant contribué à la rédaction de ce document :

- Jean-Pierre Duchesneau : version initiale, révisions
- Pierre-François Léon : révisions
- Ali Awdé : révisions