PRESENTATION DE LA METHODOLOGIE

Requêtez une base de données avec SQL

Aurelia DE INFANTI
17/01/2025

DICTIONNAIRE DES DONNÉES

Le dictionnaire de données est un référentiel centralisé qui décrit en détail la structure et le contenu des fichiers **Contrat.csv** et **Region.csv**

Il présente les informations suivantes :

- Nom de la colonne
- Type de données
- Taille du champ
- Description détaillée

	Nom des colonnes	Type de données	Taille	Contrainte	Description
CONTRAT.CSV	Contrat_ID	INT		Clé primaire	Id unique pour les contrats
	No_voie	INT			Numéro dans la voie pour l'adresse du logement assuré
	B_T_Q	CHAR	1		Indicateur éventuel de répétition pour l'adresse du logement assuré sur un caractère
	Type_de_voie	VARCHAR(20)	20		Type de voie pour l'adresse du logement assuré: rue, av (Avenue), rte (Route), ...
	Voie	VARCHAR(100)	100		Libellé de la voie pour l'adresse du logement assuré
	Code_dep_code_commune	VARCHAR(5)	5	Clé secondaire	Concaténation du code département et code commune pour avoir une clé unique
	Code_postal	VARCHAR(5)	5		Code postal pour l'adresse du logement assuré
	Surface	FLOAT			Surface du bien en m^2
	Type_local	VARCHAR(50)	50		Type de local (appartement, maison, etc.)
	Occupation	VARCHAR(50)	50		Type d'occupation (propriétaire, locataire, etc.)
	Type_contrat	VARCHAR(50)	50		Type de contrat d'assurance
	Formule	VARCHAR(50)	50		Formule d'assurance choisie
	Valeur_declaree_biens	VARCHAR(20)	20		Plage de valeurs déclarées des biens assurés (ex: '0-25000', '25000-50000')
	Prix_cotisation_mensuel	FLOAT			Prix de la cotisation mensuelle
REGION.CSV	Code_dep_code_commune	VARCHAR(5)	5	Clé primaire	Concaténation du code département et code commune pour avoir une clé unique
	reg_code	VARCHAR(2)	2		Code de la région
	reg_nom	VARCHAR(50)	50		Nom de la région
	aca_nom	VARCHAR(50)	50		Nom de l'académie
	dep_nom	VARCHAR(50)	50		Nom du département
	com_nom_maj_court	VARCHAR(50)	50		Nom de la commune en majuscules
	dep_code	VARCHAR(3)	3		Code du département
	dep_nom_num	VARCHAR(50)	50		Nom du département avec numéro

Objectifs principaux :

Comprendre la structure de la base de données

•Clarifier les relations entre les tables Contrat et Region

•Identifier les clés primaires et secondaires

•Faciliter l'interprétation et l'utilisation des données

En fournissant ces métadonnées, le dictionnaire de données permet aux utilisateurs de mieux appréhender la configuration des données, d'assurer leur intégrité et de faciliter leur exploitation dans des analyses et des requêtes SQL.

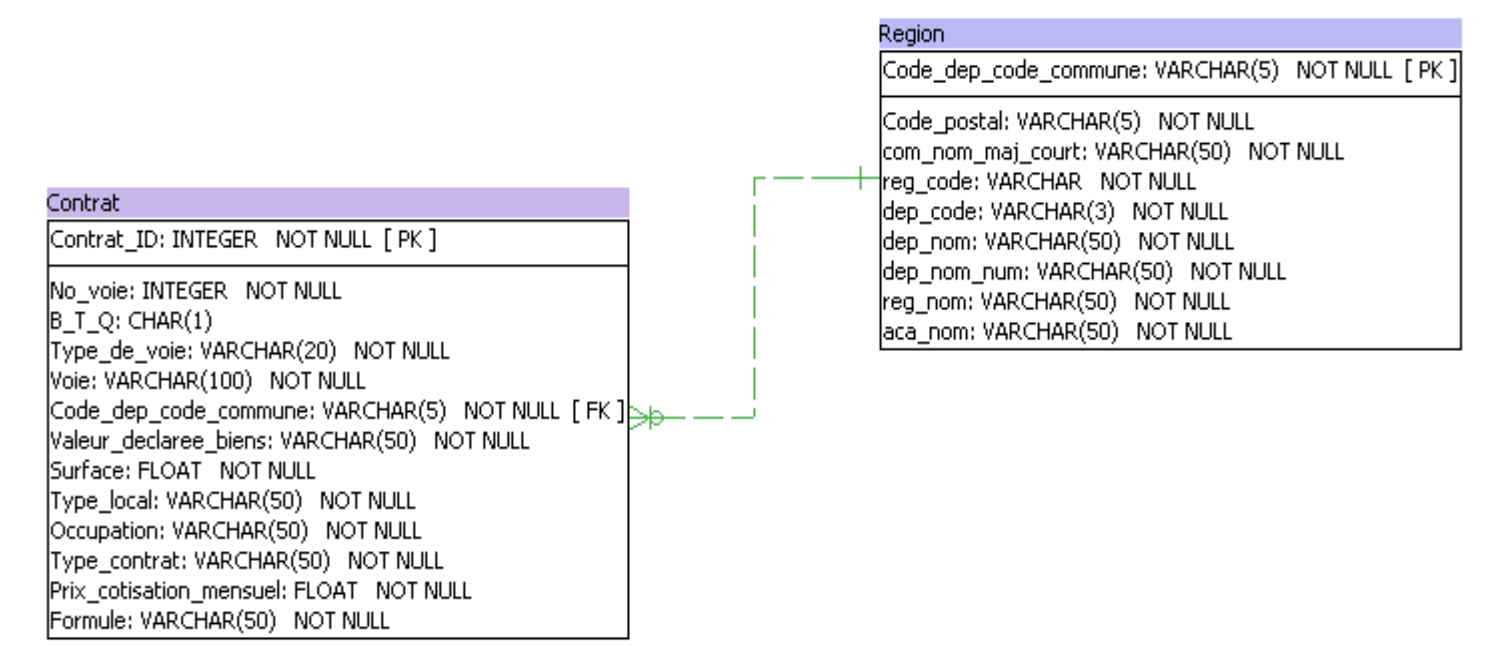
SCHEMA RELATIONNEL : Un outil essentiel

Voici le schéma relationnel des tables **Region** et **Contrat**, qui illustre leur structure et leur lien. La clé primaire de la table **Region** (*code_dep_code_commune*) est reliée à la clé étrangère de la table **Contrat** (*code_dep_code_commune*), établissant une relation de type "un-à-plusieurs".
Ce lien associe chaque contrat à une région spécifique, simplifiant l'analyse et la gestion des données.

Dans une relation un-à-plusieurs entre Region et Contrat :
Côté Region (un) :
Notation : (1,1) ou simplement (1)
Signification : Chaque contrat est associé à exactement une région.

Côté Contrat (plusieurs) :
Notation : (0,n)
Signification : Une région peut avoir zéro, un, ou plusieurs contrats.

La notation (0,n) indique :
•Minimum 0 : Une région peut exister sans aucun contrat associé.
•Maximum n : Une région peut avoir un nombre illimité de contrats.



Point de vigilance
Dep_code contient des lettres : Corse

CRÉATION BASE DE DONNÉES

TABLES CREEES

Création via SQLiteStudio

Table Region

•Contient les informations des régions et communes

•Clé primaire : code_dep_code_commune (identifiant unique)

Table Contrat

•Regroupe les informations des contrats d'assurance

•Clé primaire : contrat_id (auto-incrémentée)

•Clé étrangère : code_dep_code_commune (référence à la table Region)

Types de données

Utilisation de types adaptés :

•INTEGER pour les identifiants et numéros

•VARCHAR pour les chaînes de caractères

•FLOAT pour les nombres décimaux

```
CREATE TABLE Region (
    Code_dep_code_commune VARCHAR(5) NOT NULL,
    reg_code VARCHAR(2) NOT NULL,
    reg_nom VARCHAR(50) NOT NULL,
    aca_nom VARCHAR(50) NOT NULL,
    dep_nom VARCHAR(50) NOT NULL,
    com_nom_maj_court VARCHAR(50) NOT NULL,
    dep_code VARCHAR(3) NOT NULL,
    dep_nom_num VARCHAR(50) NOT NULL,
    PRIMARY KEY (Code_dep_code_commune)
)
```

```
CREATE TABLE Contrat (
    Contrat_ID INTEGER NOT NULL,
    No_voie INTEGER NOT NULL,
    B_T_Q CHAR(1),
    Type_de_voie VARCHAR(20) NOT NULL,
    Voie VARCHAR(100) NOT NULL,
    Code_dep_code_commune VARCHAR(5) NOT NULL,
    Code_postal VARCHAR(5) NOT NULL,
    Surface FLOAT NOT NULL,
    Type_local VARCHAR(50) NOT NULL,
    Occupation VARCHAR(50) NOT NULL,
    Type_contrat VARCHAR(50) NOT NULL,
    Formule VARCHAR(50) NOT NULL,
    Valeur_declaree_biens VARCHAR(50) NOT NULL,
    Prix_cotisation_mensuel FLOAT NOT NULL,
    PRIMARY KEY (Contrat_ID)
)
```

Relations entre les tables

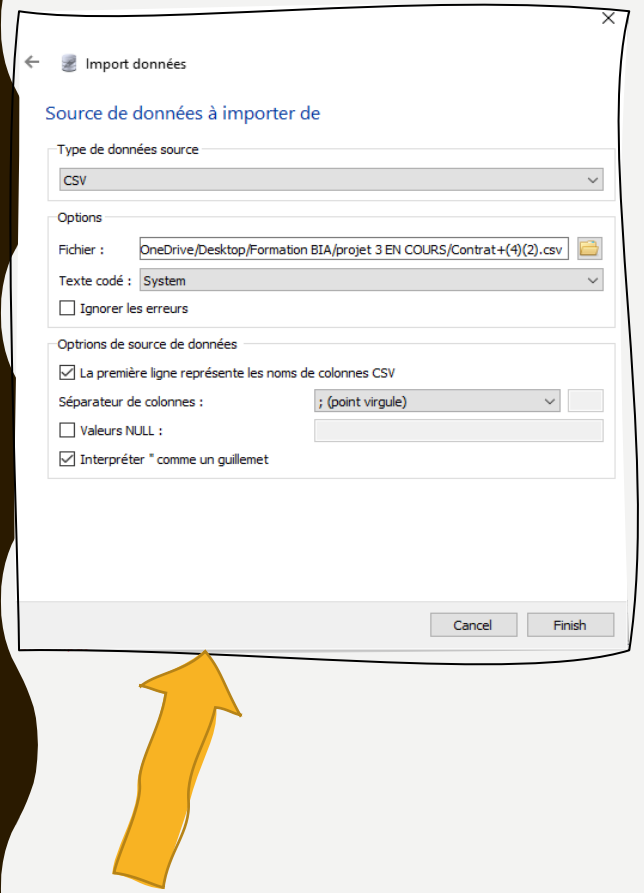
Contrainte de clé étrangère :

•Assure que chaque contrat est associé à une région spécifique

•Lie code_dep_code_commune de la table Contrat à celle de la table Region

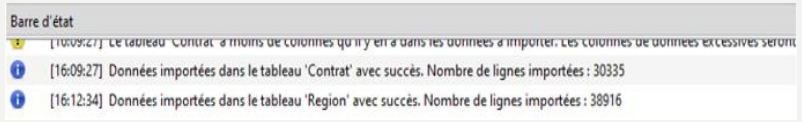
Cette structure permet une gestion efficace et cohérente des données relatives aux régions et aux contrats d'assurance.

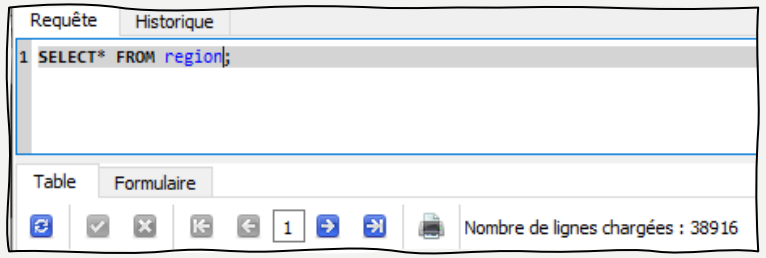
CHARGEMENT DES DONNÉES

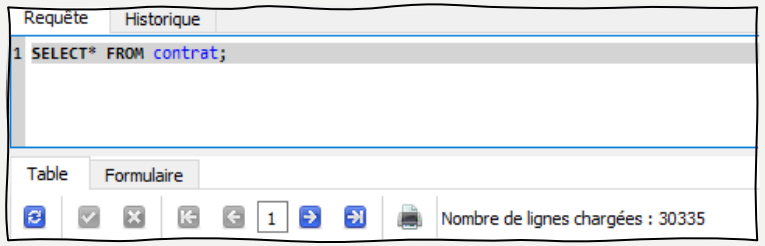


IMPORT DES FICHIERS CSV
via SQLiteStudio
Table Region
Table Contrat

VERIFICATIONS
Nombre de lignes dans :
Table Region
Table Contrat







Erreur de données

Code_dep_code_commune (97460) apparait dans contrat et non dans région. Ce qui génère un conflit quand je veux intégrer les données.

Solutions

Désactiver temporairement la vérification des clés étrangères avant l'import, puis réactivez-la après.

PRAGMA foreign_keys = OFF;
-- Effectuez votre import ici
PRAGMA foreign_keys = ON;

Points de vigilance

Type de données sources
La première ligne représente les noms de colonnes CSV
Séparateurs de colonnes

DETAILS DE LA METHODOLOGIE SUR LES REQUETES

1. Lire, comprendre la demande

- Reformuler la demande

2. Remplir un tableau

- Reformuler
- Identification des données nécessaires
- Analyse des relations entre tables
- Identification des conditions et des opérations requises
- Identification des commandes
- Pseudo-code

3. Consulter les requêtes de base

- Voir tableau ci-contre

4. Vérifier le résultat

- Vérification et débogage.

5. Refaire les requêtes

- Tous les jours tant qu'il y a des erreurs

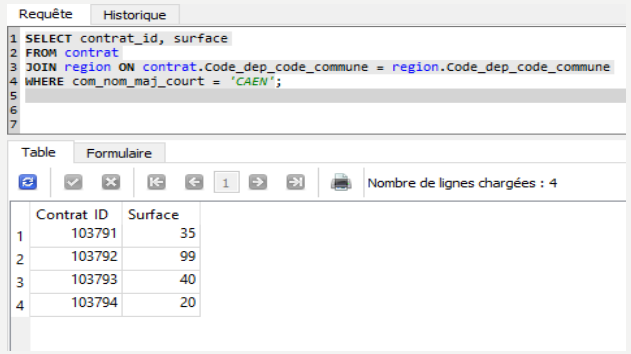
Commande	fonctions
SELECT	pour indiquer les colonnes à afficher
FROM	Pour préciser la ou les tables à interroger
WHERE	Pour filtrer les résultats selon certains critères. s'applique avant le regroupement des données (GROUP BY)
ORDER BY	Pour trier les résultats.
GROUP BY	Pour grouper les résultats par catégorie
INNER JOIN	Permettent de ne garder que les lignes qui ont une correspondance dans les deux tables
LEFT JOIN, RIGHT JOIN	Conservent toutes les lignes d'une table et les lignes correspondantes dans l'autre
HAVING	Utilisé pour filtrer des groupes basés sur des résultats d'agrégation. S'applique après le regroupement des données (GROUP BY)

Requête 1 :
Lister les n° de contrats avec leur surface pour la commune de Caen.

```
SELECT contrat_id, surface
FROM contrat, region
WHERE contrat.Code_dep_code_commune = region.Code_dep_code_commune
AND com_nom_maj_court = 'CAEN' ;

Ou

SELECT contrat_id, surface
FROM contrat
JOIN region ON contrat.Code_dep_code_commune = region.Code_dep_code_commune
WHERE com_nom_maj_court = 'CAEN';
```



Sélection des données (**SELECT**) - contrat id et la surface

Source principale des données (**FROM**) – Contrat

Jointure avec une table secondaire (**JOIN**) - grâce aux clés étrangère et primaire : Code_dep_code_commune
Filtrage des résultats

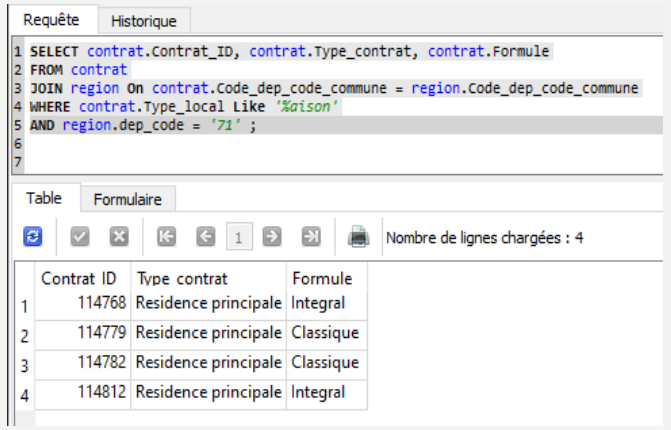
(**WHERE**) – nom de commune CAEN

Requête 2 : Lister les numéros de contrats avec le type de contrat et leur formule pour les maisons du département 71.

```
SELECT Contrat_ID, type_contrat, formule
FROM contrat, region
WHERE contrat.Code_dep_code_commune = region.Code_dep_code_commune
AND dep_code = '71'
AND LOWER (contrat.type_local) Like '%aison%';

Ou

SELECT Contrat_ID, type_contrat, formule
FROM Contrat
JOIN Region ON Contrat.Code_dep_code_commune = Region.Code_dep_code_commune
WHERE LOWER (Contrat.type_local) LIKE '%aison%'
AND Region.dep_code = 71;
```



Sélection des données **(SELECT) -** contrat id, type de contrat et formule

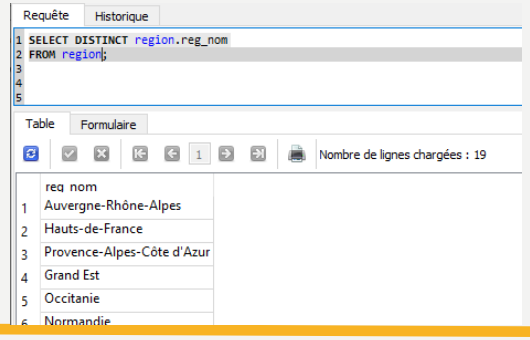
Source principale des données **(FROM) –** Contrat

Jointure avec une table secondaire **(JOIN) -** grâce aux clés étrangère et primaire : Code_dep_code_commune
Filtrage des résultats

(WHERE) – type de local maison **(AND)** code de département : 71

Requête 3 : Lister le nom des régions de France.

```
SELECT DISTINCT reg_nom
FROM Region;
```



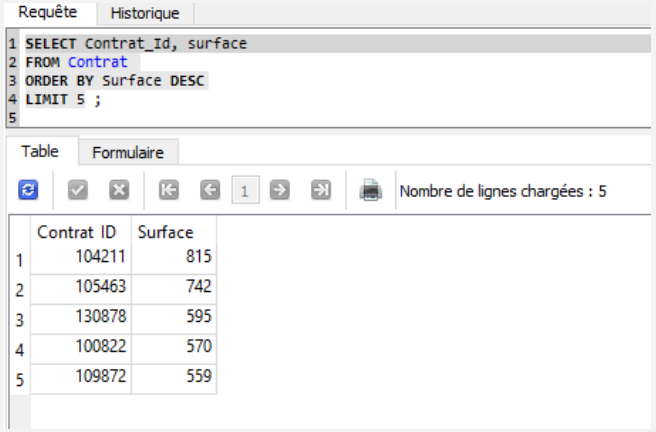
Sélection des données **(SELECT) –** nom de la région

(DISTINCT) pour ne pas qu'il y ai de doublon

Source principale des données **(FROM) -** Region

Requête 4 : Quels sont les 5 contrats ayant les surfaces les + élevées ?

```
SELECT Contrat_Id
FROM Contrat
ORDER BY Surface DESC
LIMIT 5 ;
```



Sélection des données (**SELECT**) - contrat id
Source principale des données

(**FROM**) – Contrat

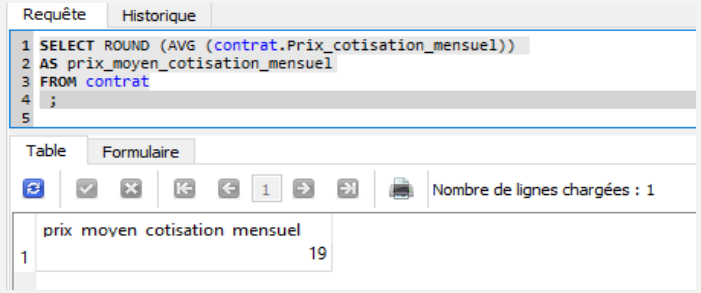
Tri des résultats (**ORDER BY**) – par surface

Tri par ordre croissant (**DESC**) – par ordre décroissant

Limitation des résultats (**LIMIT**) – résultats limités à 5

Requête 5 : Quel est le prix moyen de la cotisation mensuelle ?

```
SELECT ROUND (AVG
(Prix_cotisation_mensuel))AS
moyenne_cotisation_mensuelle
FROM Contrat ;
```



Sélection des données **(SELECT) –** prix cotisation mensuel

Agrégation **(AVG)** – calcule la moyenne du prix de cotisation mensuel

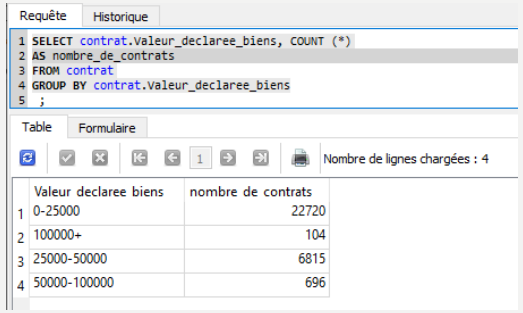
Arrondir **(ROUND)** – arrondi le résultat à l'entier le plus proche

L'alias **(AS)** moyenne_cotisation_mensuelle est utilisé pour nommer le résultat

Source principale des données **(FROM) -** Contrat

Requête 6 : Quel est le nombre de contrats pour chaque catégorie de prix de la valeur déclarée des biens

```
SELECT Valeur_declaree_biens,
COUNT (*) AS nombre_de_contrats
FROM Contrat
GROUP BY Valeur_declaree_biens;
```



```
SELECT contrat.Valeur_declaree_biens, COUNT (*)
AS nombre_de_contrats
FROM contrat
GROUP BY contrat.Valeur_declaree_biens
;
```

Table | Formulaire — Nombre de lignes chargées : 4

	Valeur declaree biens	nombre de contrats
1	0-25000	22720
2	100000+	104
3	25000-50000	6815
4	50000-100000	696

Sélection des données **(SELECT) –** valeur déclarée

Agrégation **(COUNT)** – calcule le nombre de ligne de valeur déclarées

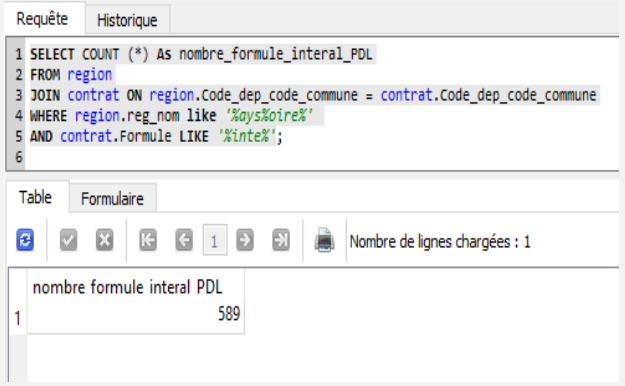
L'alias **(AS)** nombre_de_contrat est utilisé pour nommer le résultat

Source principale des données **(FROM) –** Contrat

Regroupement des données **(GROUP BY) -** regroupés selon les Valeur declaree biens

Requête 7 : Quel est le nombre de formules "intégral" sur la région Pays de la Loire ?

```
SELECT formule,
COUNT (*) as nombre_de_formule
FROM contrat
JOIN Region ON
Contrat.Code_dep_code_commune =
Region.Code_dep_code_commune
WHERE contrat.formule LIKE '%Integ%'
AND region.reg_nom LIKE '%ays%loire' ;
```



```
SELECT COUNT (*) As nombre_formule_interal_PDL
FROM region
JOIN contrat ON region.Code_dep_code_commune = contrat.Code_dep_code_commune
WHERE region.reg_nom like '%ays%loire%'
AND contrat.Formule LIKE '%inte%';
```

Table | Formulaire — Nombre de lignes chargées : 1

	nombre formule interal PDL
1	589

Sélection des données (**SELECT**) – formule

COUNT(*) : fonction d'agrégation qui compte le nombre de lignes

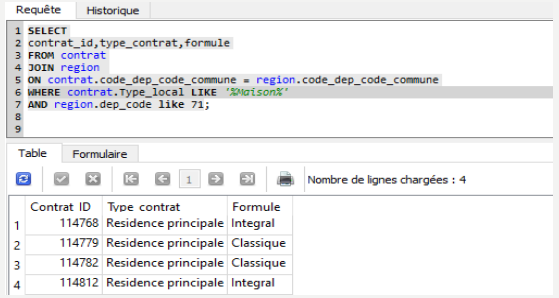
Source principale des données (**FROM**) – Contrat

Jointure avec une table secondaire (**JOIN**) - grâce aux clés étrangère et primaire : Code_dep_code_commune

Filtrage des résultats (**WHERE**) – sélectionne les formules contenant 'integ' (**AND**) selectionne les regions contenant 'ays' et 'oire'

Requête 8 : Lister les numéros de contrats avec le type de contrat et leur formule pour les maisons du département 71.

```
SELECT
contrat_id,type_contrat,formule
FROM contrat
JOIN region ON
contrat.code_dep_code_commune =
region.code_dep_code_commune
WHERE contrat.Type_local LIKE '%Maison%'
AND region.dep_code like 71;
```



```
SELECT
contrat_id,type_contrat,formule
FROM contrat
JOIN region
ON contrat.code_dep_code_commune = region.code_dep_code_commune
WHERE contrat.Type_local LIKE '%Maison%'
AND region.dep_code like 71;
```

Table | Formulaire — Nombre de lignes chargées : 4

	Contrat ID	Type contrat	Formule
1	114768	Residence principale	Integral
2	114779	Residence principale	Classique
3	114782	Residence principale	Classique
4	114812	Residence principale	Integral

Sélection des données (**SELECT**) – contrat id, type de contrat, formule

Source principale des données (**FROM**) – Contrat

Jointure avec une table secondaire (**JOIN**) - grâce aux clés étrangère et primaire : Code_dep_code_commune

Filtrage des résultats **(WHERE) –** sélectionne les code département contenant '71' (**AND**) selectionne les type de local contenant 'aison'

Requête 9 : Quelle est la surface moyenne des contrats à Paris ?

```
SELECT ROUND (AVG(surface))as surface_moyenne
FROM contrat
JOIN Region ON Contrat.Code_dep_code_commune = Region.Code_dep_code_commune
WHERE region.com_nom_maj_court LIKE 'PARIS
```

Requête | Historique

```
1 SELECT ROUND (AVG(Contrat.Surface)) AS surface_moyenne_Paris
2 FROM contrat
3 JOIN region ON region.Code_dep_code_commune = contrat.Code_dep_code_commune
4 WHERE region.com_nom_maj_court LIKE 'Paris %';
5
6
```

Table | Formulaire

Nombre de lignes chargées : 1

	surface moyenne Paris
1	52

Sélection des données **(SELECT) – surface**

(AVG) calcule la moyenne de la colonne surface
(ROUND) pour arrondir le résultat

L'alias **(AS)** surface_moyenne est utilisé pour nommer le résultat
Source principale des données **(FROM) –** Contrat

Jointure avec une table secondaire **(JOIN) -** grâce aux clés étrangère et primaire : Code_dep_code_commune

Filtrage des résultats **(WHERE) –** sélectionne le nom des communes commençant par 'Paris'

Requête 10 : Classement des 10 départements où le prix moyen de la cotisation est le plus élevé.

```
SELECT ROUND (AVG(contrat.Prix_cotisation_mensuel))AS prix_moyen_cotisation, region.dep_nom
FROM contrat
JOIN Region ON Contrat.Code_dep_code_commune = Region.Code_dep_code_commune
GROUP BY region.dep_nom
ORDER BY prix_moyen_cotisation DESC
LIMIT 10;
```

Requête | Historique

```
1 SELECT ROUND(AVG(contrat.Prix_cotisation_mensuel)) AS prix_moyen, region.dep_nom
2 FROM contrat
3 JOIN region ON region.Code_dep_code_commune = contrat.Code_dep_code_commune
4 GROUP BY region.dep_nom
5 ORDER BY prix_moyen DESC
6 LIMIT 10;
```

Table | Formulaire

Nombre de lignes chargées : 10

	prix_moyen	dep_nom
1	36	Paris
2	26	Hauts-de-Seine
3	20	Val-de-Marne
4	19	Yvelines
5	18	Rhône
6	18	Alpes-Maritimes
7	18	Ain
8	17	Seine-Saint-Denis
9	17	Haute-Savoie
10	17	Corse-du-Sud

Sélection des données (SELECT) – prix cotisation, nom département

(AVG) calcule la moyenne de la colonne prix cotisation
(ROUND) pour arrondir le résultat
L'alias **(AS)** surface_moyenne est utilisé pour nommer le résultat

Source principale des données **(FROM) –** Contrat

Jointure avec une table secondaire **(JOIN) -** grâce aux clés étrangère et primaire : Code_dep_code_commune

Regroupement des données **(GROUP BY) -** Les résultats sont regroupés par nom de département.

Tri des résultats **(ORDER BY) -** Les résultats sont triés selon prix_moyen_cotisation en ordre décroissant **(DESC) -** ordonnés selon la moyenne des cotisations par département.

Limitation des résultats **(LIMIT)** - retournés à 10

Requête 11 : Liste des communes ayant eu au moins 150 contrats.

```
SELECT region.com_nom_maj_court as commune, COUNT(*) as nombre
FROM region
JOIN Contrat ON Contrat.Code_dep_code_commune = Region.Code_dep_code_commune
GROUP BY region.com_nom_maj_court
HAVING COUNT(*) > 150
ORDER BY nombre DESC ;
```

Requête | Historique

```
SELECT COUNT(*) AS Nombre_contrat, region.com_nom_maj_court AS nom_commune
FROM region
JOIN contrat ON region.Code_dep_code_commune = contrat.Code_dep_code_commune
GROUP BY region.com_nom_maj_court
HAVING COUNT (*) >150
ORDER BY nombre_contrat;
```

Table | Formulaire — Nombre de lignes chargées : 20

	Nombre contrat	nom commune
1	159	PARIS 3
2	161	LILLE
3	163	COURBEVOIE
4	170	TOULON
5	187	TOULOUSE
6	204	PARIS 9
7	220	GRENOBLE
8	222	PARIS 14
9	252	PARIS 12
10	263	PARIS 10
11	266	PARIS 19
12	291	NANTES
13	302	BORDEAUX
14	302	PARIS 20
15	381	PARIS 11
16	387	NICE
17	394	PARIS 16
18	407	PARIS 15
19	468	PARIS 17
20	515	PARIS 18

Sélection des données **(SELECT)** – nom de commune
(COUNT) calcule Le nombre de ligne
L'alias **(AS)** nombre est utilisé pour nommer le résultat
Source principale des données **(FROM) -** region
Jointure avec une table secondaire **(JOIN) -** grâce aux clés étrangère et primaire : Code_dep_code_commune
Regroupement des données (**GROUP BY**) **-** Les résultats sont regroupés par nom de commune.
Filtrage des groupes (**HAVING**) **-** HAVING COUNT(*) > 150 filtre les résultats après le regroupement
Tri des résultats **(ORDER BY) -** Les résultats sont triés selon nombre en ordre décroissant **(DESC)**

Requête 12 : Quel est le nombre de contrats pour chaque région ?

```
SELECT DISTINCT region.reg_nom, COUNT(contrat_id) as nombre_de_contrats
FROM region
LEFT JOIN Contrat ON Contrat.Code_dep_code_commune = Region.Code_dep_code_commune
GROUP BY region.reg_nom
ORDER BY nombre_de_contrats DESC;
```

Requête | Historique

```
SELECT DISTINCT region.reg_nom AS nom_region,COUNT (contrat.Contrat_ID) AS Nombre_contrat
FROM region
LEFT JOIN contrat ON region.Code_dep_code_commune = contrat.Code_dep_code_commune
GROUP BY region.reg_nom
ORDER BY nombre_contrat;
```

Table | Formulaire — Nombre de lignes chargées : 19

	nom region	Nombre contrat
1	Collectivités d'outre-mer	0
2	Guadeloupe	0
3	Mayotte	0
4	La Réunion	8
5	Guyane	37
6	Martinique	73
7	Corse	247
8	Bourgogne-Franche-Comté	293
9	Centre-Val de Loire	598
10	Grand Est	769
11	Normandie	824
12	Bretagne	947
13	Hauts-de-France	1189
14	Pays de la Loire	1196
15	Occitanie	1609
16	Nouvelle-Aquitaine	2038
17	Auvergne-Rhône-Alpes	3042
18	Provence-Alpes-Côte d'Azur	3279
19	Ile-de-France	14177

Sélection des données **(SELECT) –** nom de region
(DISTINCT) évite les doublons
(COUNT) calcule Le nombre de ligne des contrats
L'alias **(AS)** nombre est utilisé pour nommer le résultat
Source principale des données **(FROM) -** region
Jointure avec une table secondaire **(JOIN) -** grâce aux clés étrangère et primaire : Code_dep_code_commune
Rajout de **(LEFT) -** conservent toutes les lignes d'une table et les lignes correspondantes dans l'autre
Regroupement des données **(GROUP BY) -** Les résultats sont regroupés par nom de région.
Tri des résultats **(ORDER BY) -** Les résultats sont triés selon nombre de contrats en ordre décroissant **(DESC)**