

Étape 1 : Dictionnaire des Données

Afin de mieux comprendre les types de données et les structures de nos tables, nous avons établi un dictionnaire des données décrivant les colonnes principales de chaque table.

	Nom des colonnes	Type de données	Taille	Clé	Description
CONTRAT.CSV	Contrat_ID	INT	6(pas nécessaire pour un type INT)	Clé primaire	Id unique pour les contrats
	No_voie	INT	5(pas nécessaire pour un type INT)		Numéro dans la voie pour l'adresse du logement assuré
	B_T_Q	VARCHAR	1		Indicateur éventuel de répétition pour l'adresse du logement assuré sur un caractère
	Type_de_voie	VARCHAR	7		Type de voie pour l'adresse du logement assuré: rue, av (Avenue), rte (Route), ...
	Voie	VARCHAR	30		Libellé de la voie pour l'adresse du logement assuré
	Code_dep_code_commune	VARCHAR	5	Clé secondaire	Concaténation du code département et code commune pour avoir une clé unique
	Code_postal	VARCHAR	5	Clé secondaire	Code postal pour l'adresse du logement assuré
	Surface	INT	3(pas nécessaire pour un type INT)		Surface du logement assuré, en mètres carrés
	Type_local	VARCHAR	15	Clé secondaire	Type de logement : Appartement, Maison , etc
	Occupation	VARCHAR	15		Statut de l'assuré : Locataire, Propriétaire
	Type_contrat	VARCHAR	25		Type de contrat d'assurance : Résidence principale, Résidence secondaire
	Formule	VARCHAR	15	Clé secondaire	Type de formule d'assurance : Classique , Integral
	Valeur_declaree_biens	VARCHAR	15		Plage de valeur des biens déclarés, par exemple : 0-25000
	Prix_cotisation_mensuel	INT	5		Montant mensuel de la cotisation d'assurance, exprimé en euros
REGION.CSV	Code_dep_code_commune	VARCHAR	5	Clé primaire	Concaténation du code département et code commune pour avoir une clé unique
	reg_code	INT	2	Clé secondaire	Code numérique de la région, ex: 84 pour "Auvergne-Rhône-Alpes"
	reg_nom	VARCHAR	50		Nom de la région complète, ex: "Auvergne-Rhône-Alpes"
	aca_nom	VARCHAR	50		Nom de l'académie associée à la région, ex:"Lyon"
	dep_nom	VARCHAR	50		Nom complet du département, ex: "Ain"
	com_nom_maj_court	VARCHAR	50		Nom abrégé en majuscules de la commune, ex: "AMBERIEU EN BUGEY"
	dep_code	VARCHAR	2	Clé secondaire	Code département numérique, ex: 01 pour "Ain"
	dep_nom_num	VARCHAR	50		Nom du département avec son code. ex: "Ain (01)"

Ce dictionnaire a permis de structurer nos requêtes SQL de manière plus efficace et d'éviter les erreurs de typage.

Étape 2 : Schéma Relationnel

Avant toute analyse, il était essentiel de comprendre la structure des données et leurs relations. Nous avons créé un schéma relationnel représentant les liens entre les tables contrat et région.

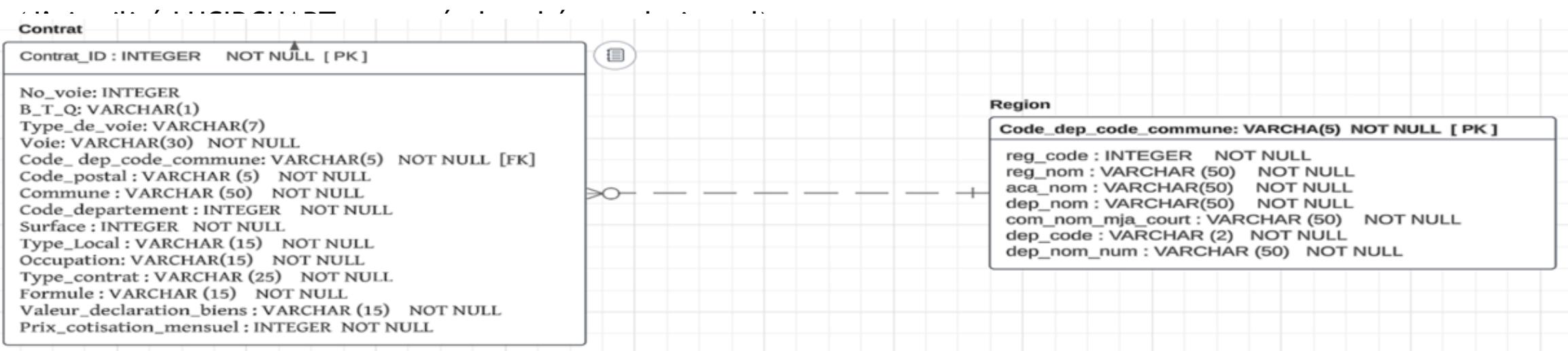
Structure du schéma relationnel :

Table contrat : Contient les informations sur chaque contrat d'assurance (surface, type de logement, prix de cotisation, etc.).

Table région : Contient les informations géographiques associées aux contrats (nom de la région, code département, etc.).

Relation : La liaison entre les deux tables se fait via Code_dep_code_commune, qui est la clé étrangère reliant contrat à région.

Ce schéma nous a permis d'optimiser les jointures et d'assurer une cohérence dans l'analyse des données.



Étape 3 : Création des Tables en SQL

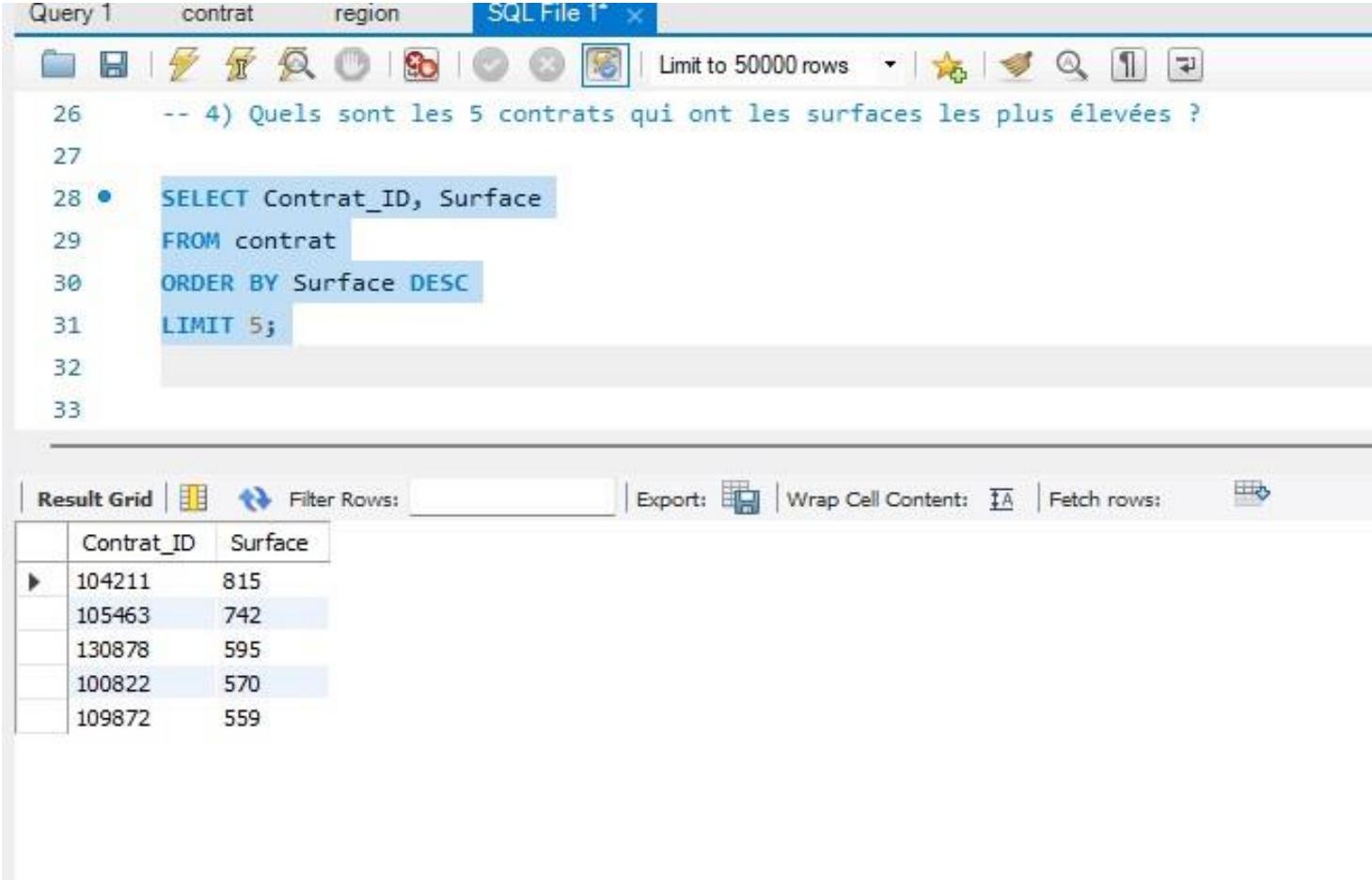
Pour assurer une base de données bien définie, nous avons utilisé SQL pour créer les tables contrat et région.

```
-- Création de la table contrat
• CREATE TABLE contrat (
    Contrat_ID INT PRIMARY KEY,
    No_voie INT,
    B_T_Q CHAR(1),
    Type_de_voie VARCHAR(7),
    Voie VARCHAR(30),
    Code_dep_code_commune VARCHAR(6),
    Code_postal VARCHAR(5),
    Surface INT,
    Type_local VARCHAR(15),
    Occupation VARCHAR(15),
    Type_contrat VARCHAR(25),
    Formule VARCHAR(15),
    Valeur_declaree_biens VARCHAR(15),
    Prix_cotisation_mensuel INT
);

-- Création de la table region
▶ CREATE TABLE region1 (
    Code_dep_code_commune VARCHAR(6) PRIMARY KEY,
    reg_code INT,
    reg_nom VARCHAR(50),
    aca_nom VARCHAR(50),
    dep_nom VARCHAR(50),
    com_nom_maj_court VARCHAR(50),
    dep_code INT,
    dep_nom_num VARCHAR(50)
);
```

Requête 4 : 5 contrats avec les surfaces les plus élevées

- Pourquoi : Utilisation de ORDER BY pour classer et de LIMIT pour récupérer les plus grandes surfaces.
- Avantage : Facile et rapide pour identifier les plus grandes valeurs.



The screenshot shows a MySQL Workbench interface with a query editor and a results grid.

Query Editor:

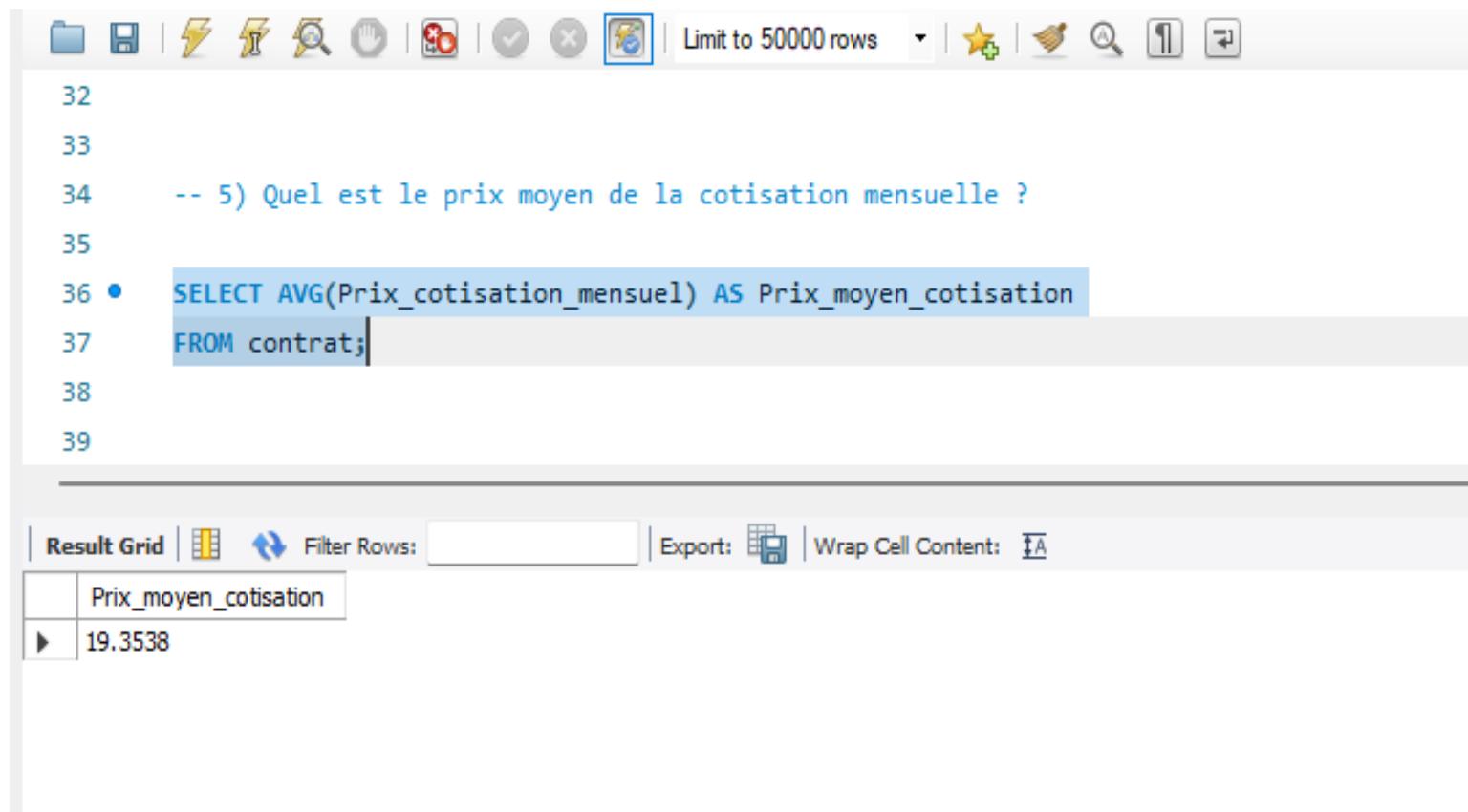
```
Query 1    contrat    region    SQL File 1* ×
26      -- 4) Quels sont les 5 contrats qui ont les surfaces les plus élevées ?
27
28 •   SELECT Contrat_ID, Surface
29     FROM contrat
30     ORDER BY Surface DESC
31     LIMIT 5;
32
33
```

Result Grid:

	Contrat_ID	Surface
▶	104211	815
	105463	742
	130878	595
	100822	570
	109872	559

Requête 5 : Prix moyen de la cotisation mensuelle

- **Pourquoi :** Utilisation de AVG() pour calculer une statistique descriptive clé.
- **Avantage :** Synthétise les données pour donner un aperçu des coûts moyens.



The screenshot shows a MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
32
33
34      -- 5) Quel est le prix moyen de la cotisation mensuelle ?
35
36 •   SELECT AVG(Prix_cotisation_mensuel) AS Prix_moyen_cotisation
37     FROM contrat;
38
39
```

Results Grid:

	Prix_moyen_cotisation
▶	19.3538

Requête 6 : Nombre de contrats par catégorie de prix

- Pourquoi : Utilisation de GROUP BY pour segmenter les données par tranche de valeur déclarée des biens.
- Avantage : Met en évidence les catégories les plus fréquentes.

The screenshot shows a MySQL Workbench interface. The top part is a SQL editor window with the following content:

```
38
39
40    -- 6) Quel est le nombre de contrats pour chaque catégorie de prix de la valeur déclarée des biens ?
41
42 • SELECT Valeur_declaree_biens, COUNT(*) AS Nombre_de_contrats
43   FROM contrat
44   GROUP BY Valeur_declaree_biens;
45
```

The line '•' before the SELECT statement indicates it is the current query being run. The bottom part is a 'Result Grid' window displaying the query results:

	Valeur_declaree_biens	Nombre_de_contrats
▶	0-25000	22603
	25000-50000	6810
	100000+	104
	50000-100000	696

Requête 7 : Formules "Integral" dans la région Pays de la Loire

- **Pourquoi :** Jointure avec filtrage sur Formule et reg_nom pour limiter les résultats.
- **Avantage :** Analyse combinée des tables pour une question spécifique.

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

- Query Tabs:** Query 1, contrat, region, SQL File 1*
- Toolbar:** Includes icons for file operations, search, and execution.
- Text Area:** Displays the SQL code for the query.

```
47      -- 7) Quel est le nombre de formules "integral" sur la région Pays de la Loire ?
48
49 • SELECT COUNT(*) AS Nombre_formules_integral
50   FROM contrat c
51   JOIN region1 r ON c.Code_dep_code_commune = r.Code_dep_code_commune
52   WHERE r.reg_nom = 'Pays de la Loire' AND c.Formule = 'INTEGRAL';
53
```
- Result Grid:** Shows the result of the query, which is a single row with one column named "Nombre_formules_integral" containing the value 586.

Requête 8 : Contrats pour les maisons du département 71

- Pourquoi : Utilisation de LEFT(Code_postal, 2) pour extraire le code département des codes postaux à 5 chiffres et filtrage basé sur le type de bien.
- Avantage : Permet de travailler directement avec les données sans modification de la structure.

The screenshot shows the MySQL Workbench interface. The top part is the SQL editor with the following query:

```
53
54
55 -- 8) Lister les numéros de contrats avec le type de contrat et leur formule pour les maisons du département 71.
56
57 • SELECT Contrat_ID, Type_contrat, Formule
58   FROM contrat
59   WHERE Type_local = 'Maison' AND LEFT(Code_postal, 2) = '71';
60
```

The bottom part is the Result Grid, displaying the following data:

Contrat_ID	Type_contrat	Formule
102369	Mise en location	Classique
102384	Residence principale	Classique
114768	Residence principale	Integral
114779	Residence principale	Classique
114782	Residence principale	Classique
114812	Residence principale	Integral

Requête 11 : Communes avec au moins 150 contrats

- Pourquoi : HAVING après regroupement pour ne conserver que les communes répondant au critère de volume.
- Avantage : Pratique pour des analyses basées sur des seuils.

The screenshot shows a MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL code for selecting communes with at least 150 contracts. The code includes a JOIN clause to link the contrat table to the region1 table based on their respective department codes.
- Result Grid:** Shows the results of the query in a tabular format. The columns are "Commune" and "Nombre_de_contrats". The data includes various cities and their contract counts, such as NICE (386), TOULOUSE (187), BORDEAUX (302), GRENOBLE (220), NANTES (291), LILLE (161), TOULON (170), PARIS 15 (407), PARIS 17 (468), PARIS 18 (515), PARIS 3 (159), PARIS 19 (266), PARIS 9 (204), PARIS 12 (252), PARIS 14 (222), PARIS 20 (302), PARIS 10 (263), COURBEV... (163), PARIS 11 (381), and PARIS 16 (394).

Commune	Nombre_de_contrats
NICE	386
TOULOUSE	187
BORDEAUX	302
GRENOBLE	220
NANTES	291
LILLE	161
TOULON	170
PARIS 15	407
PARIS 17	468
PARIS 18	515
PARIS 3	159
PARIS 19	266
PARIS 9	204
PARIS 12	252
PARIS 14	222
PARIS 20	302
PARIS 10	263
COURBEV...	163
PARIS 11	381
PARIS 16	394

