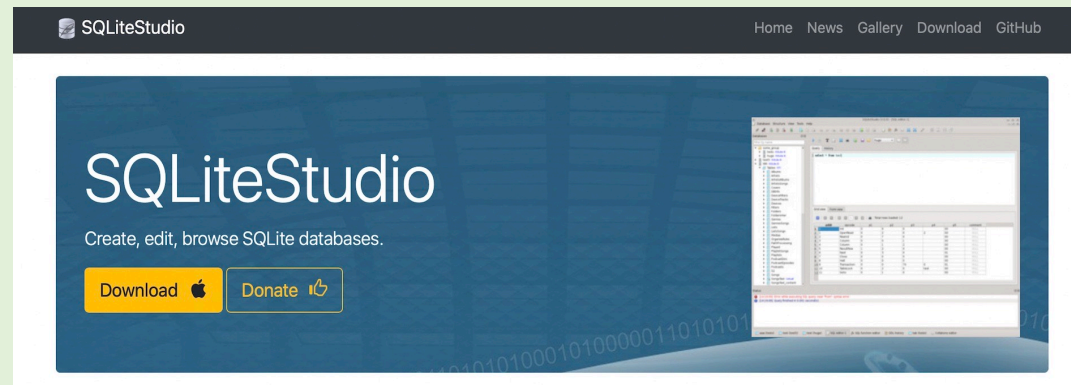




Méthodologie du projet

Requêter une base de données avec SQL

Etape 1 : installation du SGBDR



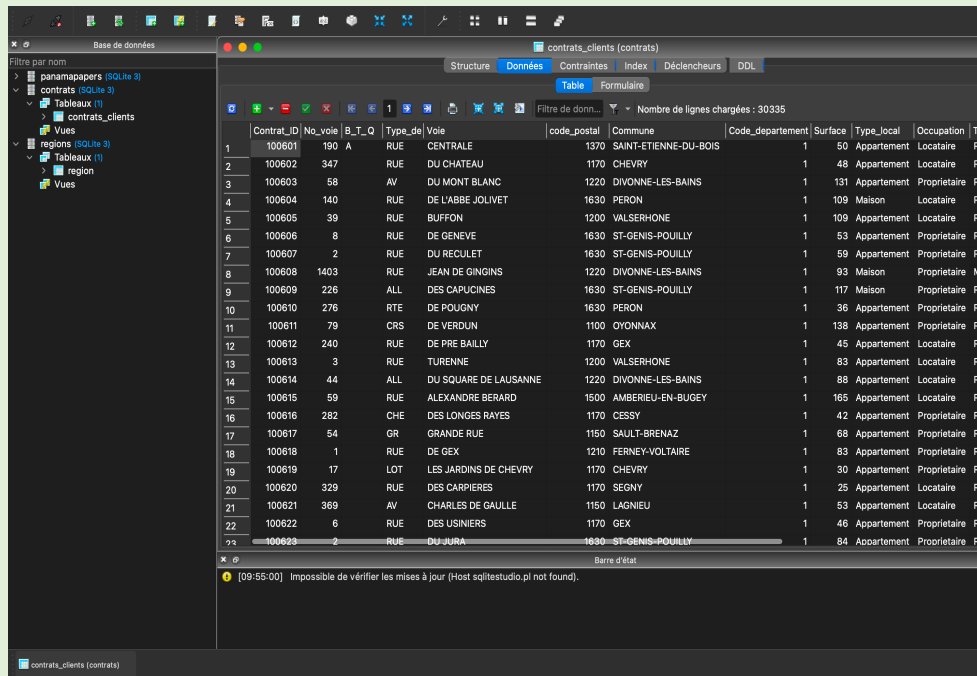
Etape 2 : chargement des données dans le SGBDR

➤ Description de la procédure d'importation des fichiers CSV :

- Ouvrir **DB Browser** for SQLite puis cliquer sur « **nouvelle base de données** »
- Enregistrer le nom de fichier sous nommé « **contrats** » ou « **regions** »
- Ouvrir **SQLite Studio**, cliquer sur « **Base de données** » puis « **ajouter une base de données** »
- Sélectionner le fichier nommé « **contrats.db** » ou « **regions.db** » précédemment enregistré avec DB Browser
- Cliquer sur l'onglet « **Outil** », « **Importer** », sélectionner le fichier « **contrats_clients.csv** » ou « **region.csv** » et cliquer sur « **Done** »

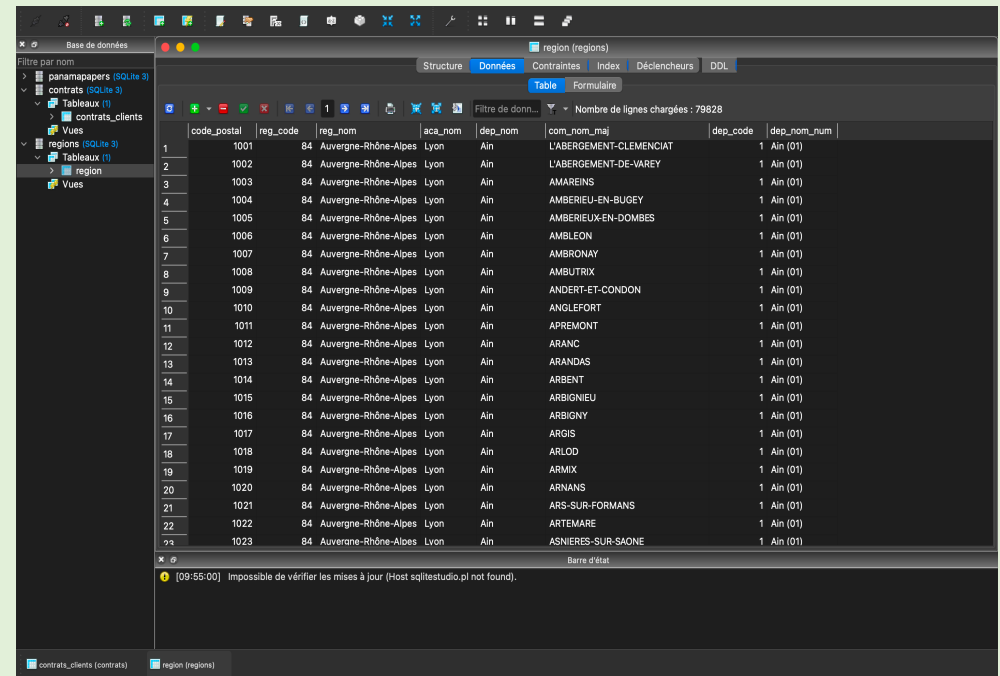
Etape 2 : Chargement des données dans le SGBDR

TABLE CONTRATS_CLIENTS



	Contrat_ID	No_voie	B_T_Q	Type_de	Voie	code_postal	Commune	Code_departement	Surface	Type_local	Occupation	Ty
1	100601	190	A	RUE	CENTRALE	1370	SAINT-ETIENNE-DU-BOIS	50	Appartement	Locataire	Re	
2	100602	347		RUE	DU CHATEAU	1170	CHEVRY	48	Appartement	Locataire	Re	
3	100603	58		AV	DU MONT BLANC	1220	DIVONNE-LES-BAINS	131	Appartement	Propriétaire	Re	
4	100604	140		RUE	DE L'ABBE JOLIVET	1630	PERON	109	Maison	Locataire	Re	
5	100605	39		RUE	BUFFON	1200	VALSERHONE	109	Appartement	Locataire	Re	
6	100606	8		RUE	DE GENEVE	1630	ST-GENIS-POUILLY	53	Appartement	Propriétaire	Re	
7	100607	2		RUE	DU RECULET	1630	ST-GENIS-POUILLY	59	Appartement	Propriétaire	Re	
8	100608	1403		RUE	JEAN DE GINGINS	1220	DIVONNE-LES-BAINS	93	Maison	Propriétaire	Mi	
9	100609	226		ALL	DES CAPUCINES	1630	ST-GENIS-POUILLY	117	Maison	Propriétaire	Re	
10	100610	276		RTE	DE POUIGNY	1630	PERON	36	Appartement	Propriétaire	Re	
11	100611	79		CRS	DE VERDUN	1100	OYONNAX	138	Appartement	Propriétaire	Re	
12	100612	240		RUE	DE PRE BAILLY	1170	GEX	45	Appartement	Locataire	Re	
13	100613	3		RUE	TURENNE	1200	VALSERHONE	83	Appartement	Locataire	Re	
14	100614	44		ALL	DU SQUARE DE LAUSANNE	1220	DIVONNE-LES-BAINS	88	Appartement	Locataire	Re	
15	100615	59		RUE	ALEXANDRE BERARD	1500	AMBERIEU-EN-BUGEY	165	Appartement	Locataire	Re	
16	100616	282		CHE	DES LONGES RAYES	1170	CESSY	42	Appartement	Propriétaire	Re	
17	100617	54		GR	GRANDE RUE	1150	SAULT-BRENAZ	68	Appartement	Propriétaire	Re	
18	100618	1		RUE	DE GEX	1210	FERNEY-VOLTAIRE	83	Appartement	Propriétaire	Re	
19	100619	17		LOT	LES JARDINS DE CHEVRY	1170	CHEVRY	30	Appartement	Propriétaire	Re	
20	100620	329		RUE	DES CARRIERES	1170	SENY	25	Appartement	Locataire	Re	
21	100621	369		AV	CHARLES DE GAULLE	1150	LAGNIEU	53	Appartement	Locataire	Re	
22	100622	6		RUE	DES USINIERS	1170	GEX	46	Appartement	Propriétaire	Re	
23	100623	2		RUE	DU JURA	1630	ST-GENIS-POUILLY	84	Appartement	Propriétaire	Re	

TABLE REGION



	code_postal	reg_code	reg_nom	aca_nom	dep_nom	com_nom_maj	dep_code	dep_nom_num
1	1001	84	Auvergne-Rhône-Alpes	Lyon	Ain	L'ABERGEMENT-CLEMENCIAT	1	Ain (01)
2	1002	84	Auvergne-Rhône-Alpes	Lyon	Ain	L'ABERGEMENT-DE-VAREY	1	Ain (01)
3	1003	84	Auvergne-Rhône-Alpes	Lyon	Ain	AMAREINS	1	Ain (01)
4	1004	84	Auvergne-Rhône-Alpes	Lyon	Ain	AMBERIEU-EN-BUGEY	1	Ain (01)
5	1005	84	Auvergne-Rhône-Alpes	Lyon	Ain	AMBERIEUX-EN-DOBES	1	Ain (01)
6	1006	84	Auvergne-Rhône-Alpes	Lyon	Ain	AMBLEON	1	Ain (01)
7	1007	84	Auvergne-Rhône-Alpes	Lyon	Ain	AMBRONAY	1	Ain (01)
8	1008	84	Auvergne-Rhône-Alpes	Lyon	Ain	AMBUTRIX	1	Ain (01)
9	1009	84	Auvergne-Rhône-Alpes	Lyon	Ain	ANDERT-ET-CONDON	1	Ain (01)
10	1010	84	Auvergne-Rhône-Alpes	Lyon	Ain	ANGLEFORT	1	Ain (01)
11	1011	84	Auvergne-Rhône-Alpes	Lyon	Ain	APREMONT	1	Ain (01)
12	1012	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARANC	1	Ain (01)
13	1013	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARANDAS	1	Ain (01)
14	1014	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARBENT	1	Ain (01)
15	1015	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARBIGNIEU	1	Ain (01)
16	1016	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARBIGNY	1	Ain (01)
17	1017	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARGIS	1	Ain (01)
18	1018	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARLOD	1	Ain (01)
19	1019	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARMIX	1	Ain (01)
20	1020	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARNANS	1	Ain (01)
21	1021	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARS-SUR-FORMANS	1	Ain (01)
22	1022	84	Auvergne-Rhône-Alpes	Lyon	Ain	ARTEMARE	1	Ain (01)
23	1023	84	Auvergne-Rhône-Alpes	Lyon	Ain	ASNIERES-SUR-SAONE	1	Ain (01)

Etape 3: Exploration des données

The screenshot shows the Microsoft Excel interface with the following data table:

	B	C	D	E	F	G	
1							
2	Nom du champs	Type de données	Taille (nb caractères max)	Contrainte	Description	Table	
3	Contrat_ID	INT	10	Clé primaire	Id unique pour les contrats	Contrat	
4	No_voie	INT	10		Numéro dans la voie pour l'adresse du logement assuré		
5	B_T_Q	TEXT	10		Indicateur éventuel de répétition pour l'adresse du logement assuré		
6	Type_de_voie	TEXT	10		Type de voie pour l'adresse du logement assuré: rue, av (Avenue), rte (Route), ...		
7	Voie	TEXT	50		Libellé de la voie pour l'adresse du logement assuré		
8	Code_postal	INT	10		Code postal pour l'adresse du logement assuré		
9	Commune	TEXT	50		Libellé de la commune de l'adresse du logement		
10	Code_département	TEXT	20		Code du département pour l'adresse du logement		
11	Surface	NUMERIC	5		Surface du logement		
12	Type_local	TEXT	30		Type de logement (Maison, Appartement)		
13	Occupation	TEXT	30		Type d'occupation (Locataire ou Propriétaire)		
14	Type_contrat	TEXT	50		Type de contrat (Résidence principale, secondaire ou mise en location)		
15	Formule	TEXT	50		Type de formule choisi par l'assuré (Integral ou Classique)		
16	Valeur_declare_mobilier	TEXT	50		Montant déclaré par l'assuré. Cette valeur sert de base à l'indemnisation en cas de sinistre.		
17	Prix_cotisation_mensuel	INT	50		La cotisation d'assurance correspond à la somme que doit payer l'assuré chaque mois,		
18	code_postal	INT	30	Clé primaire	code commune (referentiel-geographique-francais, source www.data.gouv.fr)		Region
19	reg_code	INT	20		Code région		
20	reg_nom	TEXT	50		Libellé région		
21	aca_nom	TEXT	50		Libellé de l'académie		
22	dep_nom	TEXT	50		Libellé du département		
23	com_nom_maj	TEXT	50		Libellé de la commune en majuscule		
24	dep_code	TEXT	50		Code département		
25	dep_nom_num	TEXT	50		Libellé du département et code		

Accueil

Insertion

Dessin

Mise en page

Formules

Données

Révision

Affichage

Dites-le-nous

Partager

Coller

Calibri (Corps)

12

A^a

A^v

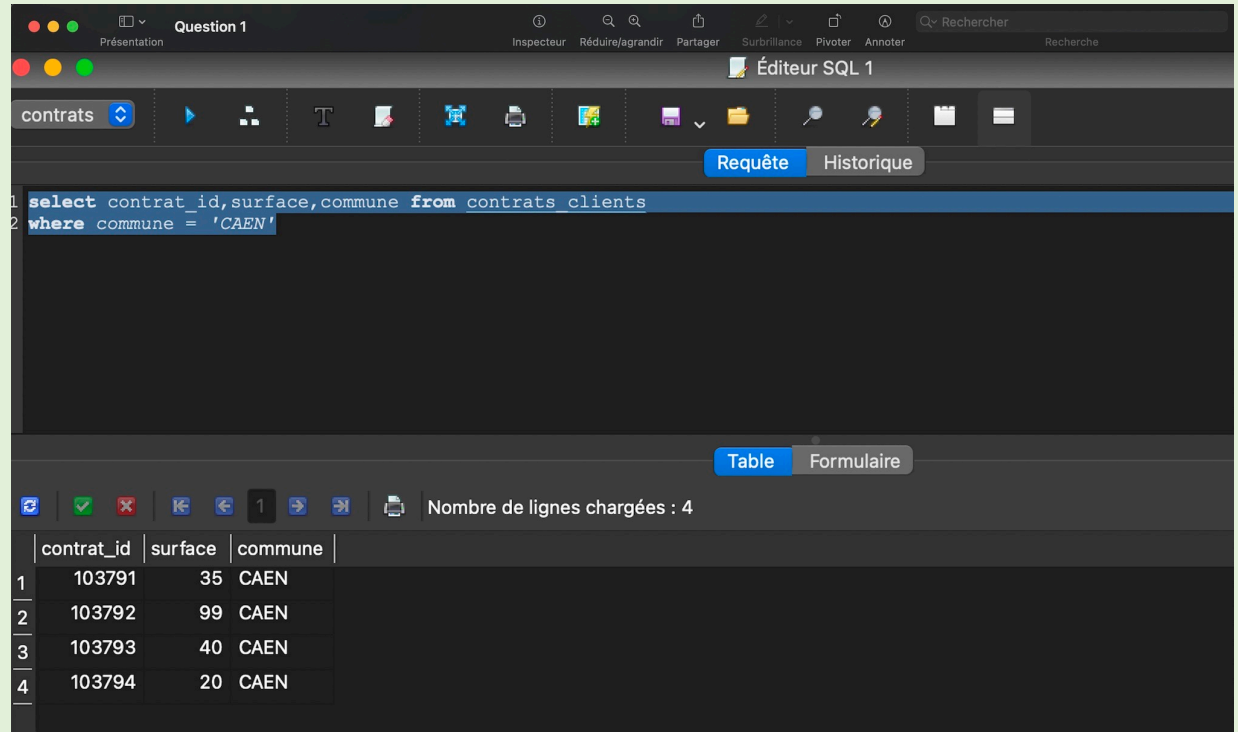
</

Etape 4 : analyse des trois demandes

<div> <div> <div>▲</div> <div>▼</div> </div> <div>✕</div> <div>✓</div> <div>f_x</div> </div> Lister les numéros de contrats (contrat_ID)				
A	B	C	D	E
	Besoin	Informations recherchées	Clauses	Requete
1	Lister les numéros de contrats (contrat_ID) avec leur surface pour la commune de Caen	Numéros de contrats (contrat_ID), surface, commune Caen	select from where	<pre>select contrat_id, surface, commune from contrats_clients where commune = 'CAEN'</pre>
2	Lister les numéros de contrats (contrat_ID), avec le type de contrat et leur formule pour les maisons du département de la Saône-et-Loire (Département 71)	Numéros de contrats (contrat_ID), type de contrat et leur formule, type local (maison), code département 71	select from where And	<pre>select contrat_ID, type_contrat, formule, type_local, code_departement from contrats_clients where code_departement = '71' and type_local = 'Maison'</pre>
3	Lister le nom des régions de France	régions de France (reg_nom)	select distinct	<pre>select distinct reg_nom from region</pre>

Etape 5 : mise en pratique des notions fondamentales

- La **projection** programmée dans le **SELECT** permet d'extraire **trois colonnes** (contrat_id, surface, commune). Cette commande sert à sélectionner une ou plusieurs colonnes de la table.
- L'instruction **FROM** sert à spécifier dans quelle table nous souhaitons chercher les données. Ici, c'est la **table contrats_clients** qui nous intéresse.
- La commande **WHERE** est une **opération de restriction** qui nous permet de filtrer les lignes que nous souhaitons afficher.



The screenshot shows a SQL editor window titled "Question 1". The query editor contains the following SQL code:

```
1 select contrat_id,surface,commune from contrats_clients
2 where commune = 'CAEN'
```

Below the query editor, there are tabs for "Requête" (selected) and "Historique". At the bottom, there is a "Table" tab and a "Formulaire" tab. The "Table" tab displays the results of the query in a table format:

	contrat_id	surface	commune
1	103791	35	CAEN
2	103792	99	CAEN
3	103793	40	CAEN
4	103794	20	CAEN

Below the table, it indicates "Nombre de lignes chargées : 4".

Requête 1: lister les numéros de contrats avec leur surface pour la commune de Caen.

Etape 5 : mise en pratique des notions fondamentales

- **Projection** : **SELECT** contrat_ID, surface
- **Clause d'ordonnement** : **ORDER BY DESC** qui permet d'effectuer un **tri descendant** de la surface.
- La clause **LIMIT** restreint le résultat de la requête aux **5 contrats** ayant les surfaces les plus élevées.

The screenshot shows a database application interface. On the left, a tree view displays the database structure: 'panamapapers (SQLite 3)' containing 'contrats (SQLite 3)', which has a table 'contrats_clients' with 15 columns. The columns listed are: Contrat_ID, No_voie, B_T_Q, Type_de_voie, Voie, Code_postal, Commune, Code_departement, Surface, Type_local, Occupation, Type_contrat, Formule, Valeur_declaree_biens, and Prix_cotisation_mensuel. The main window is titled 'Éditeur SQL 1' and shows a SQL query:

```
1 select contrat_id, surface
2 from contrats_clients
3 order by surface desc
4 limit 5
```

 Below the query editor, a table view shows the results of the query. The table has two columns: 'contrat_id' and 'surface'. The results are as follows:

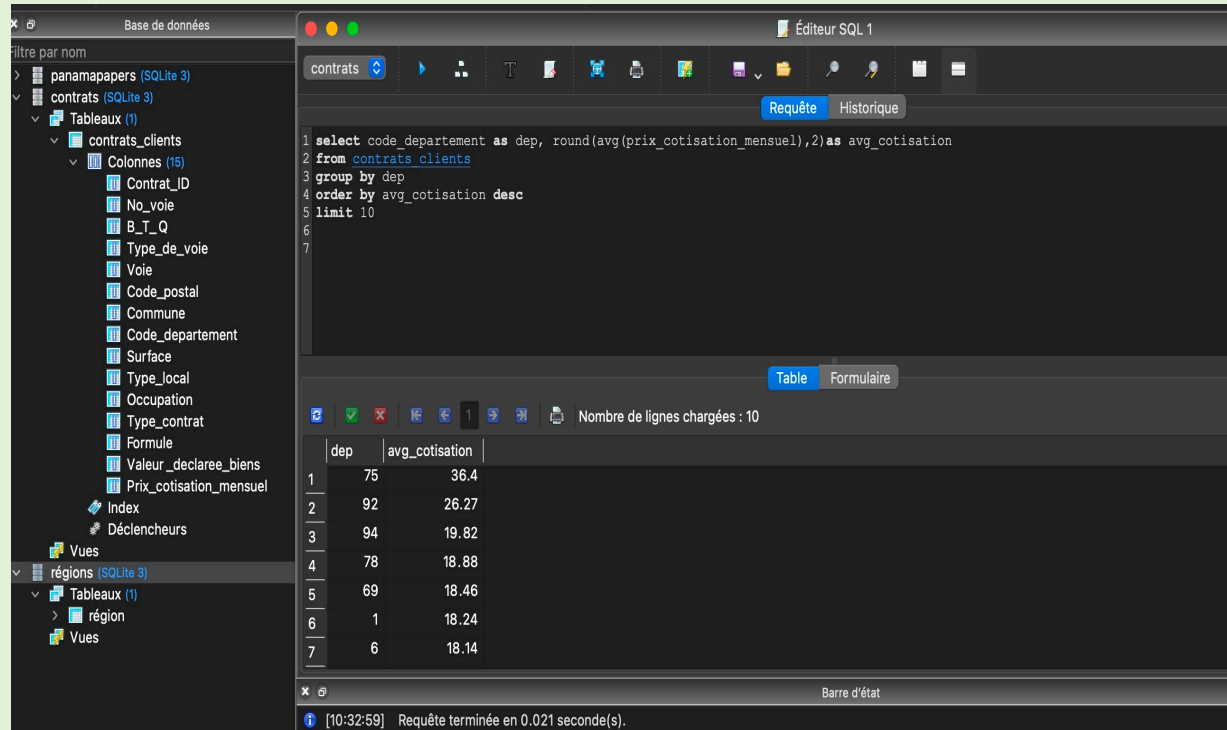
	contrat_id	surface
1	104211	815
2	105463	742
3	130878	595
4	100822	570
5	109872	559

At the bottom, a status bar indicates 'Nombre de lignes chargées : 5' and a log shows several successful query executions.

Requête 6 : quels sont les 5 contrats qui ont les surfaces les plus élevées ?

Etape 5 : mise en pratique des notions fondamentales

- **Projection** : **SELECT** code_département, prix_cotisation_mensuel
- **Alias** : **as** dep, **as** avg_cotisation
- **Fonction d'aggrégation (ou regroupement)**: **AVG**(prix_cotisation_mensuel) nous permet de calculer le prix moyen de la cotisation.
- **Fonction numérique** : **ROUND()** permet d'arrondir la moyenne de la cotisation avec 2 chiffres après la virgule.
- **Clause de regroupement** : **GROUP BY** permet de regrouper les départements selon la moyenne calculée.
- **Clause d'ordonnement** : **ORDER BY DESC** permet d'effectuer un **tri descendant** du prix moyen des cotisations.
- La clause **LIMIT** restreint le résultat de la requête aux **10 départements** ayant les cotisations moyennes les plus élevées.



The screenshot shows a SQL editor interface with a database schema on the left and a query editor on the right. The query editor contains the following SQL code:

```
1 select code_département as dep, round(avg(prix_cotisation_mensuel),2) as avg_cotisation
2 from contrats_clients
3 group by dep
4 order by avg_cotisation desc
5 limit 10
6
7
```

Below the query editor, the results are displayed in a table format. The table has two columns: 'dep' and 'avg_cotisation'. The results are as follows:

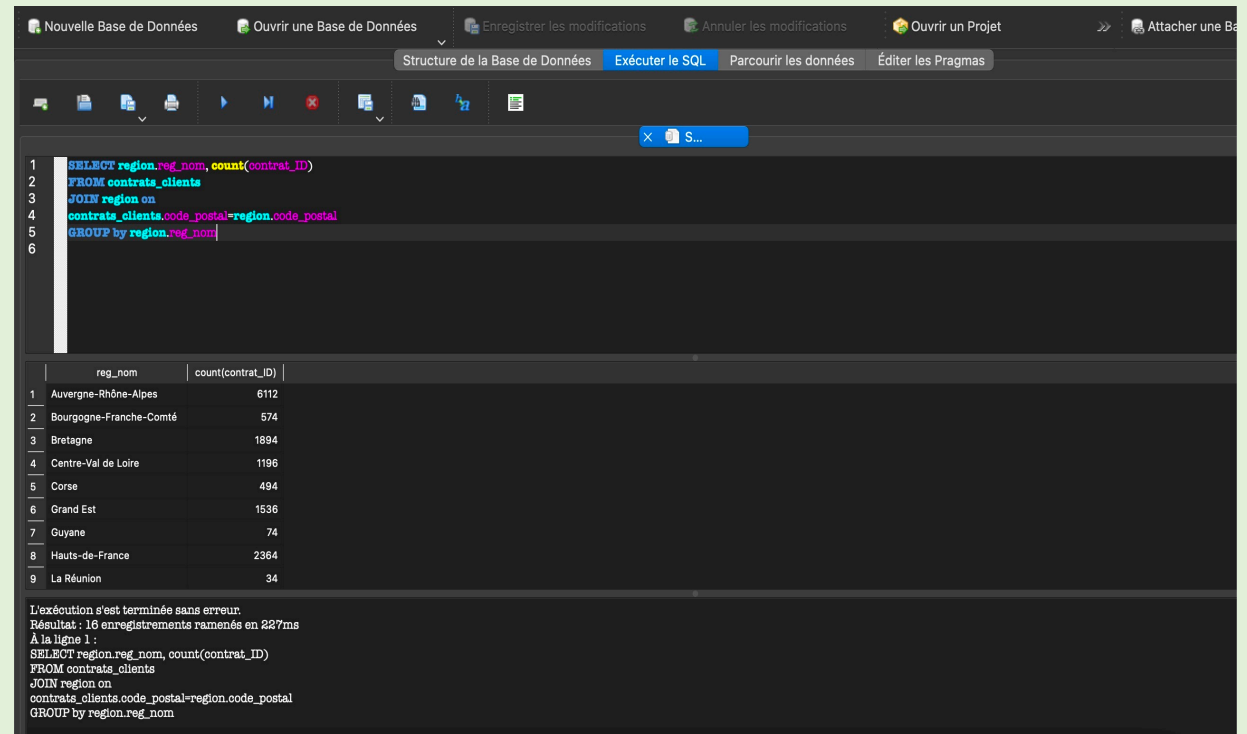
	dep	avg_cotisation
1	75	36.4
2	92	26.27
3	94	19.82
4	78	18.88
5	69	18.46
6	1	18.24
7	6	18.14

The status bar at the bottom indicates that the query was terminated in 0.021 seconds.

Requête 9 : classement des 10 départements où le prix moyen de la cotisation est le plus élevé.

Etape 5 : mise en pratique des notions fondamentales

- **Projection** : `SELECT region.reg_nom, count(contrat_ID)`
- L'instruction **From contrats_clients** est la table dans laquelle nous cherchons les données.
- **JOIN ON** permet d'effectuer une jointure naturelle de la **table region** à partir de l'**attribut code_postal** commun aux deux tables.
- **Clause d'ordonnement** : **ORDER BY DESC** permet d'effectuer un **tri descendant** du nom de chaque région.



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```
1 SELECT region.reg_nom, count(contrat_ID)
2 FROM contrats_clients
3 JOIN region on
4 contrats_clients.code_postal=region.code_postal
5 GROUP by region.reg_nom
6
```

The results pane displays a table with two columns: `reg_nom` and `count(contrat_ID)`. The data is as follows:

reg_nom	count(contrat_ID)
1 Auvergne-Rhône-Alpes	6112
2 Bourgogne-Franche-Comté	574
3 Bretagne	1894
4 Centre-Val de Loire	1196
5 Corse	494
6 Grand Est	1536
7 Guyane	74
8 Hauts-de-France	2364
9 La Réunion	34

Below the table, the IDE shows the execution status: "L'exécution s'est terminée sans erreur. Résultat : 16 enregistrements ramenés en 227ms. À la ligne 1 :". The query text is repeated below this status.

Requête 12 : quel est le nombre de contrat pour chaque région ?