

Requête 1 : Lister les numéros de contrats (contrat\_ID) avec leur surface pour la commune de Caen.

The screenshot shows the pgAdmin 4 interface. The top menu bar includes File, Object, Tools, Edit, View, Window, and Help. The main window displays a SQL query in the 'Query' tab. The query is as follows:

```
1 SELECT c.contrat_id, c.surface
2 FROM contrat AS c
3 INNER JOIN region AS r
4   ON c.code_dep_code_commune = r.code_dep_code_commune
5 WHERE r.com_nom_maj_court = 'CAEN';
6
7
```

Below the query editor, the 'Data Output' tab shows the results of the query. The results are displayed in a table with two columns: 'contrat\_id [PK] integer' and 'surface integer'. There are four rows of data:

	contrat_id [PK] integer	surface integer
1	103791	35
2	103792	99
3	103793	40
4	103794	20

At the bottom of the interface, a status bar indicates 'Total rows: 4' and 'Query complete 00:00:00.081'.

### Interprétation de la Requête 1

Cette requête permet d'extraire les **contrats d'assurance** associés à la **commune de Caen**.

Elle effectue une **jointure interne** **INNER JOIN** entre les tables **contrat** et **region**, en se basant sur la colonne **code\_dep\_code\_commune**, afin de relier chaque contrat à sa région. **ON** définit la condition de jointure entre les 2 tables.

**INNER JOIN** + **ON** va :

- conserver uniquement les lignes où le code **code\_dep\_code\_commune** est présent dans les deux tables
- et associer correctement les données des deux tables

Ensuite, elle applique un filtre **WHERE** pour **ne conserver que les contrats situés dans la commune de "CAEN"** **com\_nom\_maj\_court = 'CAEN'**.

Les informations retournées sont les **identifiants de contrat** **contrat\_id** ainsi que la **surface assurée** **surface**, comme demandé dans l'énoncé.

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

The screenshot shows the pgAdmin 4 interface. The top bar indicates the user is connected to 'Assurance/postgres@PostgreSQL 17'. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT c.contrat_id, c.type_contrat, c.formule
2 FROM contrat AS c
3 INNER JOIN region AS r
4   ON c.code_dep_code_commune = r.code_dep_code_commune
5 WHERE c.type_local = 'Maison' AND r.dep_code = '71';
6
```

Below the query editor, the 'Data Output' tab shows the results of the query in a table with 4 rows and 3 columns: **contrat\_id** (integer), **type\_contrat** (character varying), and **formule** (character varying).

	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

The status bar at the bottom indicates 'Total rows: 4' and 'Query complete 00:00:00.177'.

### Interprétation de la requête 2

La requête SQL permet d'identifier l'ensemble des contrats d'assurance habitation ayant pour **type de logement une maison**, et localisés dans le **département 71 (Saône-et-Loire)**.

Pour cela, elle effectue une **jointure interne**

**INNER JOIN** entre la table contrat et la table region, en se basant sur le code de la commune (code\_dep\_code\_commune) **ON** définit la condition de jointure entre les 2 tables.

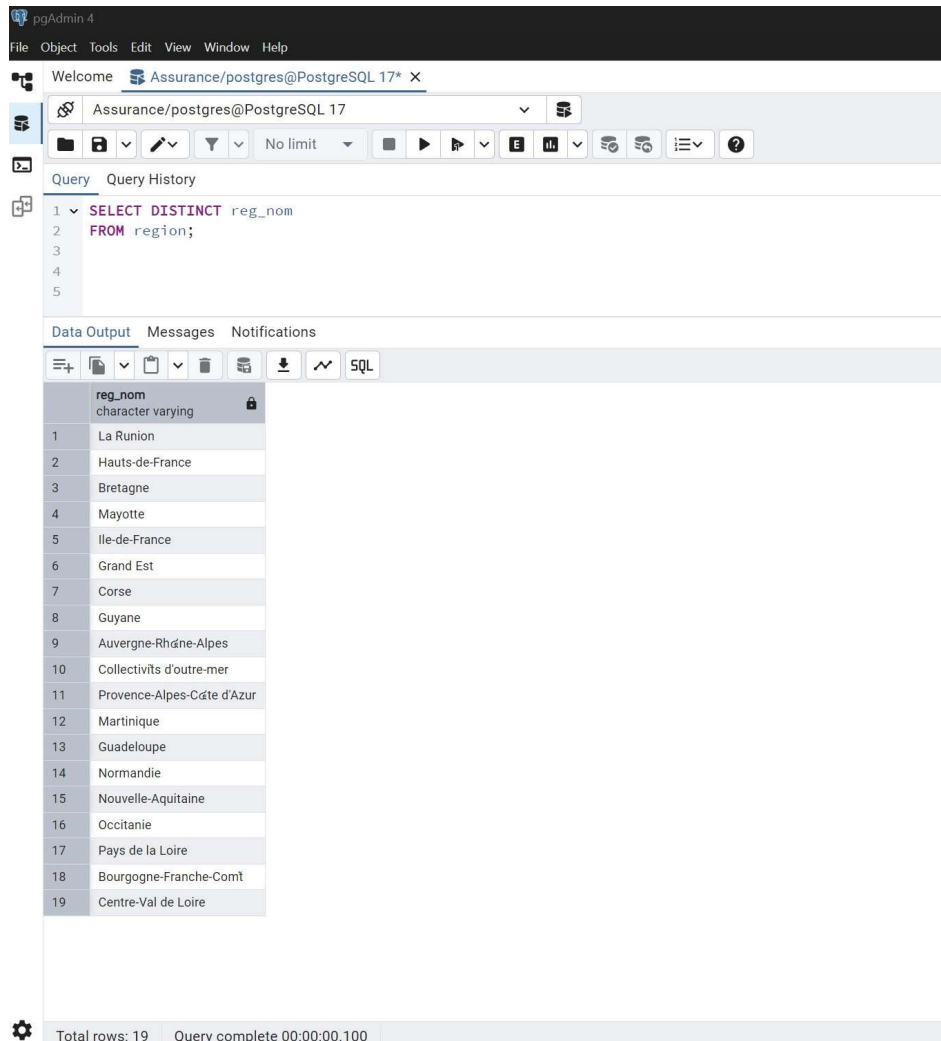
**INNER JOIN** + **ON** va :

- conserver uniquement les lignes où le code code\_dep\_code\_commune est présent dans les deux tables
- et associer correctement les données des deux tables

Ensuite, avec **WHERE** elle filtre les résultats afin de ne conserver que les contrats correspondant à des maisons (*type\_local = 'Maison'*) situées dans le département 71 (*dep\_code = '71'*) grâce au **AND**

Enfin, les informations affichées incluent : l'identifiant du contrat, le type de contrat, la formule choisie, le type de logement, ainsi que le code du département.

Requête 3 : Lister le nom des régions de France.



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL query:

```
1 SELECT DISTINCT reg_nom
2 FROM region;
3
4
5
```

The query results are displayed in the Data Output tab, showing a list of 19 regions. The status bar at the bottom indicates 'Total rows: 19' and 'Query complete 00:00:00,100'.

reg_nom
La Runion
Hauts-de-France
Bretagne
Mayotte
Ile-de-France
Grand Est
Corse
Guyane
Auvergne-Rhône-Alpes
Collectivités d'outre-mer
Provence-Alpes-Côte d'Azur
Martinique
Guadeloupe
Normandie
Nouvelle-Aquitaine
Occitanie
Pays de la Loire
Bourgogne-Franche-Comté
Centre-Val de Loire

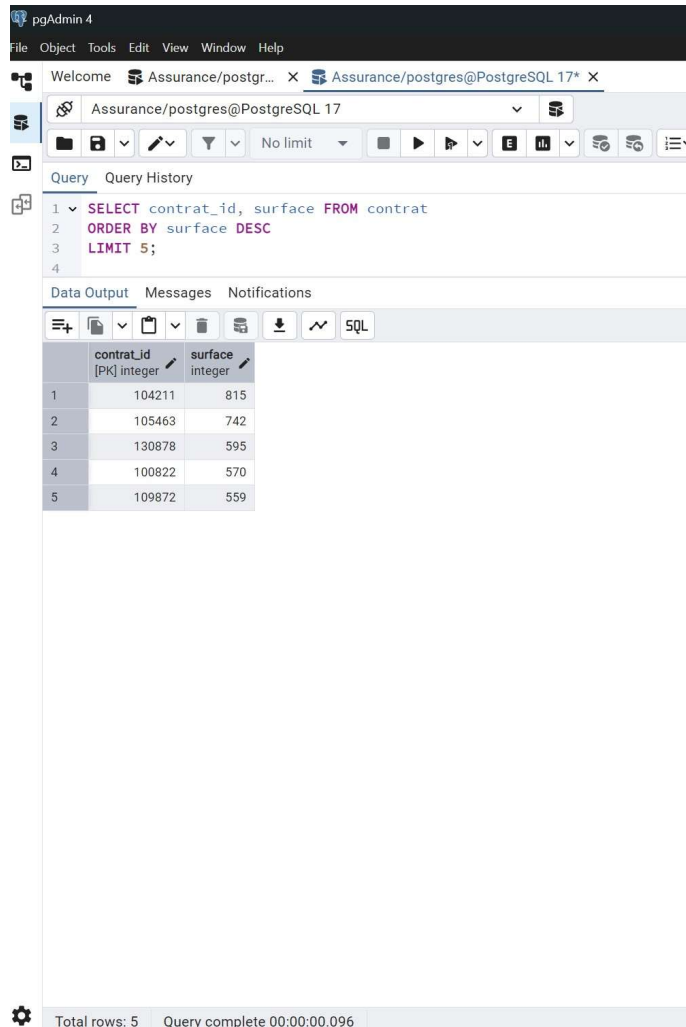
### Interprétation de la Requête 3

Cette requête permet d'obtenir la **liste des régions françaises** présentes dans la base de données.

Elle interroge la table région et retourne **les noms uniques des régions** (reg\_nom).

L'utilisation DISTINCT permet d'éliminer les doublons éventuels et de ne garder qu'une **occurrence par région**.

Requête 4 : Quels sont les 5 contrats qui ont les surfaces les plus élevées ?



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1 SELECT contrat_id, surface FROM contrat
2 ORDER BY surface DESC
3 LIMIT 5;
```

The results are displayed in a table with the following data:

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

At the bottom of the interface, a status bar indicates: Total rows: 5 Query complete 00:00:00.096

## Interprétation de la Requête 4

### Objectif de la requête :

L'objectif est d'**identifier les 5 contrats présentant les surfaces les plus importantes** au sein de la table .

`ORDER BY surface DESC` : tri décroissant sur la surface, de la plus grande à la plus petite.

`LIMIT 5` : limitation aux 5 premières lignes du classement, soit les 5 plus grandes surfaces.

Cette analyse permet de mettre en évidence les contrats les plus significatifs en termes de superficie.

La requête renvoie les 5 contrats ayant les **surfaces maximales** enregistrées dans la base de données.

Cette requête permet d'obtenir rapidement une **vue synthétique et pertinente** sur les contrats les plus importants en termes de surface.

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

Query Query History

```
1 SELECT ROUND (AVG(prix_cotisation_mensuel),2) AS prix_moyen
2 FROM contrat;
3
4
5
```

Data Output Messages Notifications

	prix_moyen numeric
1	19.33

### Interprétation de la Requête 5

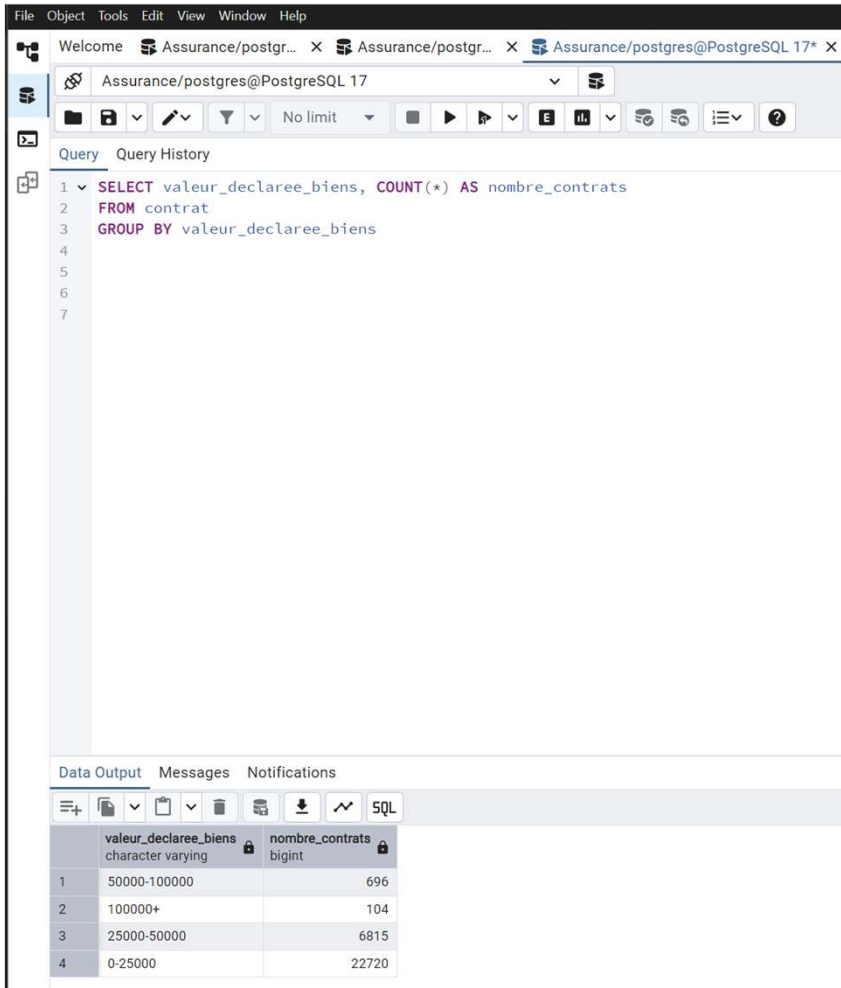
#### Objectif de la requête :

L'objectif est de **déterminer le prix moyen de la cotisation mensuelle**.

**ROUND** (**AVG** (prix\_cotisation\_mensuel),2): utilise la fonction d'agrégation appliquée sur la colonne correspondant à la cotisation mensuelle dans la table Contrat. Le **ROUND** permet d'arrondi à 2 décimales après la virgule.

Cette analyse permet de calculer la moyenne de prix de la cotisation mensuelle pour tous les contrats enregistrés.

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



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT valeur_declaree_biens, COUNT(*) AS nombre_contrats
2 FROM contrat
3 GROUP BY valeur_declaree_biens
```

The results are displayed in a table with two columns: `valeur_declaree_biens` (character varying) and `nombre_contrats` (bigint). The data is as follows:

	valeur_declaree_biens	nombre_contrats
1	50000-100000	696
2	100000+	104
3	25000-50000	6815
4	0-25000	22720

### Interprétation de la Requête 6

#### Objectif de la requête :

L'objectif est de **déterminer le nombre de contrats pour chaque catégorie de la colonne prix valeur déclarée des biens**.

`COUNT (*)` permet de compter le nombre de contrat,  
`GROUP BY` permet de regrouper par catégorie.

Elle interroge la colonne prix de valeur de la table Contrat puis regroupe les résultats par catégorie.

Cette analyse permet de compter le nombre de contrats pour chaque catégorie.

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

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT COUNT(*) AS nombre_formules_integral
2 FROM contrat AS c
3 INNER JOIN region AS r
4   ON c.code_dep_code_commune = r.code_dep_code_commune
5 WHERE c.formule = 'Integral'
6   AND r.reg_nom = 'Pays de la Loire';
```

The result is displayed in the 'Data Output' tab, showing a single row with the count of 589.

nombre_formules_integral
589

### Interprétation de la Requête 7

#### Objectif de la requête :

L'objectif est de **déterminer le nombre de contrats ayant la formule « Integral » et localisés dans la région « Pays de Loire »**.

Pour cela, elle effectue une **jointure interne**

INNER JOIN entre la table contrat et la table region, en se basant sur le code de la commune (code\_dep\_code\_commune). **ON** définit la condition de jointure entre les 2 tables.

**INNER JOIN** + **ON** va :

- conserver uniquement les lignes où le code code\_dep\_code\_commune est présent dans les deux tables
- et associer correctement les données des deux tables

**WHERE** est le filtre qui permet de ne retenir que les lignes où la formule est « Integral » et où la région est « Pays de la Loire ».

Elle interroge la colonne prix de valeur de la table Contrat puis regroupe les résultats par catégorie.

Cette analyse permet de calculer le nombre total de contrats ayant les 2 conditions.

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

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

```
1 SELECT c.contrat_id, c.type_contrat, c.formule, c.type_local, r.dep_code
2 FROM contrat AS c
3 INNER JOIN region AS r
4   ON c.code_dep_code_commune = r.code_dep_code_commune
5 GROUP BY c.contrat_id, c.type_contrat, c.formule, c.type_local, r.dep_code
6 HAVING type_local = 'Maison' AND dep_code = '71';
7
8
9
10
11
```

Below the query editor, the 'Data Output' tab displays the results of the query in a table:

	contrat_id integer	type_contrat character varying	formule character varying	type_local character varying	dep_code character varying
1	114768	Residence principale	Integral	Maison	71
2	114779	Residence principale	Classique	Maison	71
3	114782	Residence principale	Classique	Maison	71
4	114812	Residence principale	Integral	Maison	71

### Interprétation de la requête 8 (exactement la même que la requête 2)

La requête permet d'identifier l'ensemble des contrats d'assurance habitation ayant pour **type de logement une maison**, et localisés dans le **département 71 (Saône-et-Loire)**.

Afin de voir si le résultat est identique, J'ai fait l'usage d'une autre possibilité en requête,

Pour cela, elle effectue une **jointure interne**

**INNER JOIN** entre la table contrat et la table region, en se basant sur le code de la commune (code\_dep\_code\_commune). ). **ON** définit la condition de jointure entre les 2 tables.

**INNER JOIN + ON** va :

- conserver uniquement les lignes où le code code\_dep\_code\_commune est présent dans les deux tables
- et associer correctement les données des deux tables

**GROUP BY** permet de regrouper par colonne,

Ensuite, avec **HAVING** elle filtre les résultats afin de ne conserver que les contrats correspondant à des maisons (type\_local = 'Maison') situées dans le département 71 (dep\_code = '71') grâce au **AND**.

Enfin, les informations affichées incluent : l'identifiant du contrat, le type de contrat, la formule choisie, le type de logement, ainsi que le code du département.



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

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

```
1 SELECT ROUND (AVG(c.surface), 2) AS surface_moyenne_paris
2 FROM contrat AS c
3 INNER JOIN region AS r
4 ON c.code_dep_code_commune = r.code_dep_code_commune
5 WHERE r.com_nom_maj_court LIKE 'PARIS%';
```

The query is executed, and the results are displayed in the Data Output tab:

surface_moyenne_paris
51.77

## Interprétation de la Requête 9

### Objectif de la requête :

L'objectif est de **déterminer la surface moyenne des contrats à Paris**.

Pour cela, elle effectue une **jointure interne**

**INNER JOIN** entre la table contrat et la table region, en se basant sur le code de la commune (code\_dep\_code\_commune). **ON** définit la condition de jointure entre les 2 tables.

**INNER JOIN + ON** va :

- conserver uniquement les lignes où le code code\_dep\_code\_commune est présent dans les deux tables
- et associer correctement les données des deux tables

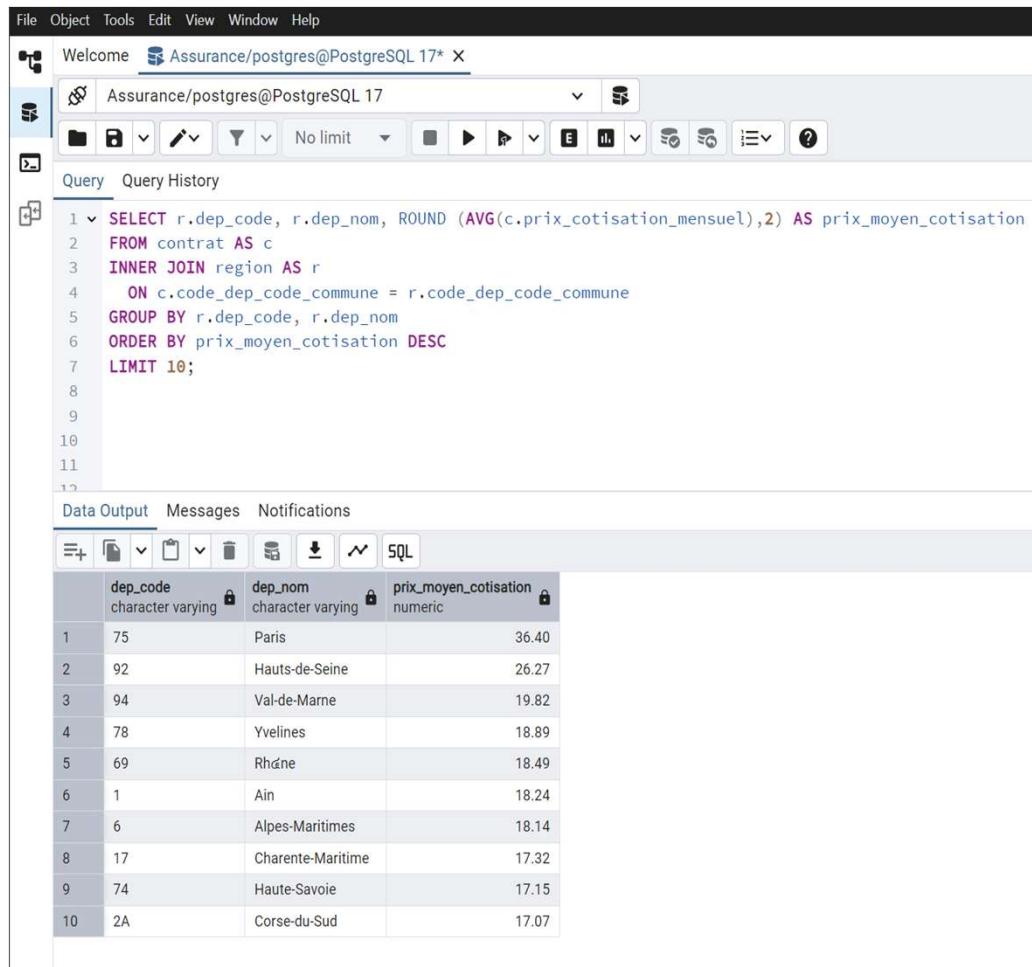
**ROUND (AVG (c.surface),2)** : utilise la fonction d'agrégation appliquée sur la colonne correspondant à la cotisation mensuelle dans la table Contrat. Le **ROUND** permet d'arrondi à 2 décimales après la virgule.

**WHERE** est le filtre qui permet de ne retenir que les lignes avec une surface et où la ville est Paris.

**LIKE** permet ainsi de regrouper tous les arrondissements de la ville de Paris en utilisant un filtre et l'utilisation « % » signifie n'importe quel texte après,

Cette analyse permet de déterminer la surface moyenne des contrats sur tous les arrondissements de Paris,

Requête 10 : Classement des 10 départements où le prix moyen de la cotisation est la plus élevée,



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT r.dep_code, r.dep_nom, ROUND (AVG(c.prix_cotisation_mensuel),2) AS prix_moyen_cotisation
2 FROM contrat AS c
3 INNER JOIN region AS r
4   ON c.code_dep_code_commune = r.code_dep_code_commune
5 GROUP BY r.dep_code, r.dep_nom
6 ORDER BY prix_moyen_cotisation DESC
7 LIMIT 10;
```

The results are displayed in a table with the following columns: dep\_code, dep\_nom, and prix\_moyen\_cotisation. The table contains 10 rows, representing the top 10 departments by average monthly premium.

dep_code	dep_nom	prix_moyen_cotisation
75	Paris	36.40
92	Hauts-de-Seine	26.27
94	Val-de-Marne	19.82
78	Yvelines	18.89
69	Rhône	18.49
1	Ain	18.24
6	Alpes-Maritimes	18.14
17	Charente-Maritime	17.32
74	Haute-Savoie	17.15
2A	Corse-du-Sud	17.07

## Interprétation de la Requête 10

### Objectif de la requête :

L'objectif est d'**obtenir les 10 départements ayant le prix moyen de la cotisation le plus élevé**.

Pour cela, elle effectue une **jointure interne**

**INNER JOIN** entre la table contrat et la table region, en se basant sur le code de la commune (code\_dep\_code\_commune). **ON** définit la condition de jointure entre les 2 tables.

**INNER JOIN** + **ON** va :

- conserver uniquement les lignes où le code code\_dep\_code\_commune est présent dans les deux tables
- et associer correctement les données des deux tables

**ROUND (AVG (c.prix\_cotisation\_mensuel),2)** : utilise la fonction d'agrégation appliquée sur la colonne correspondant à la cotisation mensuelle dans la table Contrat. Le **ROUND** permet d'arrondi à 2 décimales après la virgule.

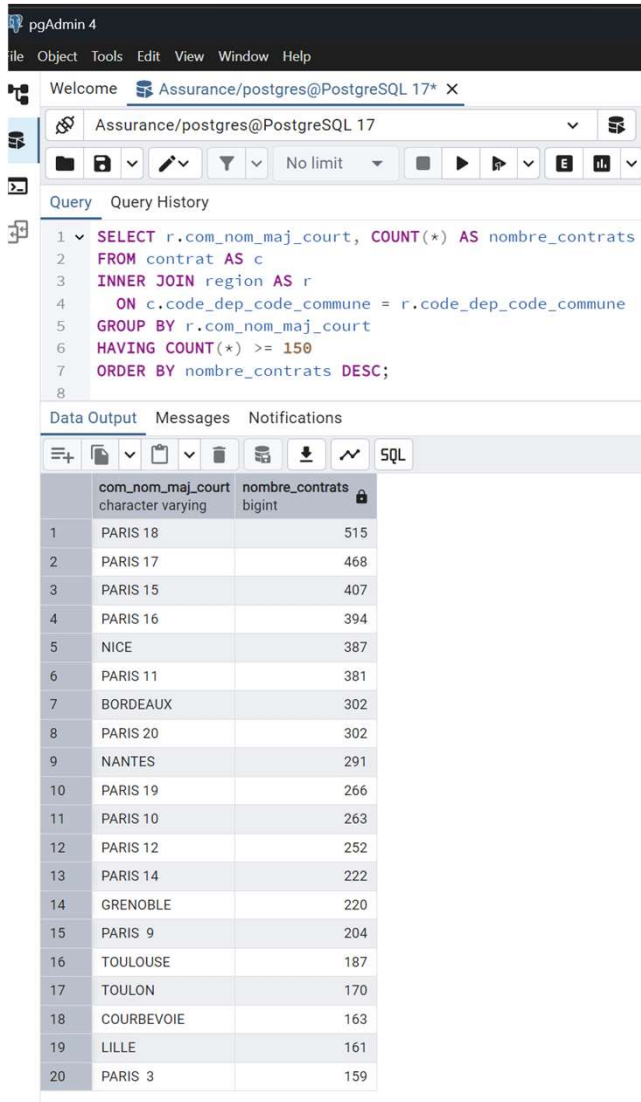
**GROUP BY** permet de regrouper par catégorie.

**ORDER BY** + **DESC** : tri décroissant sur le prix de la cotisation, de la plus grande à la plus petite.

**LIMIT 10** : limitation aux 10 premières lignes du classement, soit les 10 cotisations les plus élevées.

Cette analyse permet de lister les 10 premiers départements avec la cotisation mensuelle la plus élevée.

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



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL query:

```
1 SELECT r.com_nom_maj_court, COUNT(*) AS nombre_contrats
2 FROM contrat AS c
3 INNER JOIN region AS r
4   ON c.code_dep_code_commune = r.code_dep_code_commune
5 GROUP BY r.com_nom_maj_court
6 HAVING COUNT(*) >= 150
7 ORDER BY nombre_contrats DESC;
```

The query results are displayed in a table with two columns: `com_nom_maj_court` (character varying) and `nombre_contrats` (bigint). The results are sorted in descending order of the number of contracts.

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

### Interprétation de la Requête 11

#### Objectif de la requête :

L'objectif est d'**obtenir la liste des communes ayant eu au moins 150 contrats**,

Pour cela, elle effectue une **jointure interne**

`INNER JOIN` entre la table `contrat` et la table `region`, en se basant sur le code de la commune (`code_dep_code_commune`). `ON` définit la condition de jointure entre les 2 tables.

`INNER JOIN` + `ON` va :

- conserver uniquement les lignes où le code `code_dep_code_commune` est présent dans les deux tables
- et associer correctement les données des deux tables

`COUNT (*)` permet de compter le nombre de contrat,

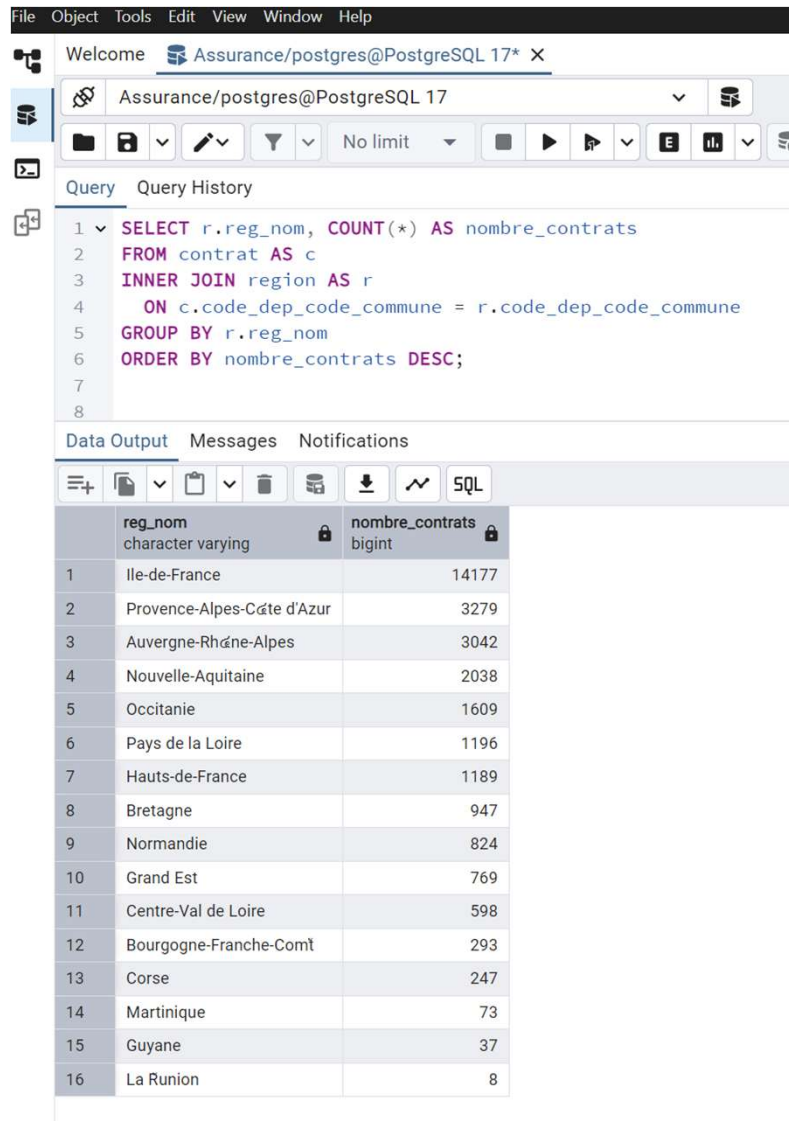
`GROUP BY` permet de regrouper par catégorie.

`HAVING COUNT (*) >= 150` : filtre pour ne garder que les communes ayant au moins 150 contrats et plus

`ORDER BY + DESC` : trie les résultats du plus grand au plus petit nombre de contrats,

Cette analyse permet d'obtenir la liste des communes ayant au moins 150 contrats et plus.

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



The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 SELECT r.reg_nom, COUNT(*) AS nombre_contrats
2 FROM contrat AS c
3 INNER JOIN region AS r
4 ON c.code_dep_code_commune = r.code_dep_code_commune
5 GROUP BY r.reg_nom
6 ORDER BY nombre_contrats DESC;
```

The results are displayed in a table with two columns: **reg\_nom** (character varying) and **nombre\_contrats** (bigint). The results are ordered by the number of contracts in descending order.

reg_nom	nombre_contrats
Ile-de-France	14177
Provence-Alpes-Côte d'Azur	3279
Auvergne-Rhône-Alpes	3042
Nouvelle-Aquitaine	2038
Occitanie	1609
Pays de la Loire	1196
Hauts-de-France	1189
Bretagne	947
Normandie	824
Grand Est	769
Centre-Val de Loire	598
Bourgogne-Franche-Comté	293
Corse	247
Martinique	73
Guyane	37
La Réunion	8

## Interprétation de la Requête 12

### Objectif de la requête :

L'objectif est **de calculer le nombre de contrats par région**.

Pour cela, elle effectue une **jointure interne**

**INNER JOIN** entre la table contrat et la table region, en se basant sur le code de la commune (code\_dep\_code\_commune). **ON** définit la condition de jointure entre les 2 tables.

**INNER JOIN + ON** va :

- conserver uniquement les lignes où le code code\_dep\_code\_commune est présent dans les deux tables
- et associer correctement les données des deux tables

**COUNT (\*)** permet de compter le nombre de contrat.

**GROUP BY** permet de regrouper par catégorie.

**ORDER BY + DESC** : trie les résultats du plus grand au plus petit nombre de contrats,

Cette analyse permet d'obtenir le nombre de contrats par région dans l'ordre du plus grand nombre au plus petit nombre de contrats.