



Création et utilisation de la base de données

Melchiori Manuel



Laplace Immo

Contexte du projet

DATAImmo

- Nettoyage, creation et gestion d'une BDD immobiliere



La stratégie de sauvegarde et la conformité RGPD

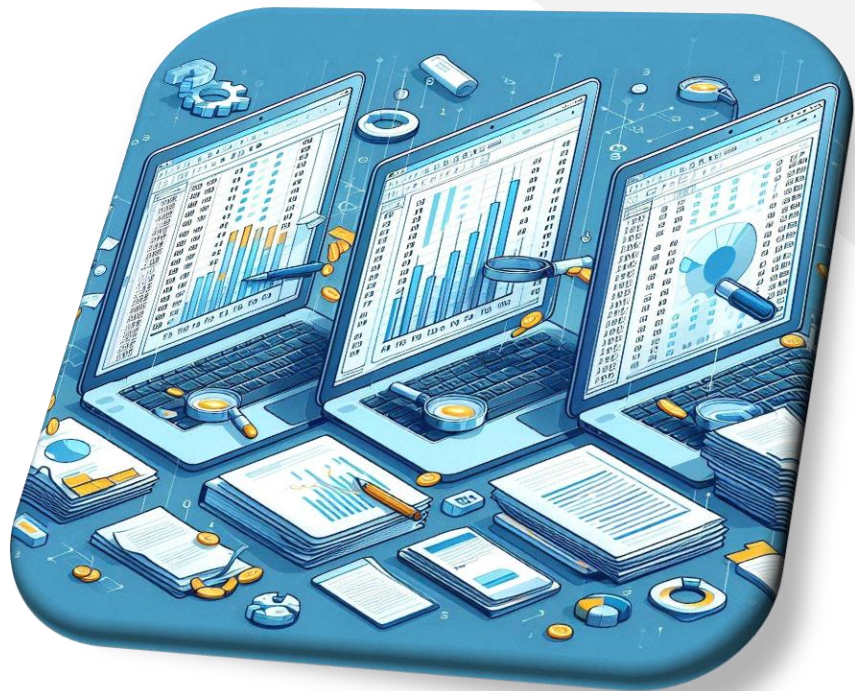
- Les données ont été nettoyées ainsi:
- Suppression des noms prénoms
- Suppression des adresses

Le rgpd est respecté, les données personnelles
Ne sont pas importées dans la BDD



Les données initiales

- Valeurs-foncières:
Incluant date, valeurs, descriptif du bien
- fr-esr-referentiel-géographique:
Contiens les données géographiques
- donnees_communes:
Contient les données communales, population etc



L'extrait du dictionnaire des données

DICTIONNAIRE DES DONNÉES - Valeurs foncières						
CODE	SIGNIFICATION	TYPE	LONGUEUR	NATURE	REGLE DE GESTION	REGLE DE CALCUL
Id_bien	ID du bien	Integer	NC	calculer	Clé primaire	incrément de 1
Date_mutation	Date de signature de l'acte	Date	NC	Élémentaire	aaaa/mm/jj	NC
Valeur_fonciere	La valeur foncière est le prix net vendeur. La TVA est incluse.	Float	NC	Élémentaire	Ne doit pas être nul	NC
id_codedep_codecom	Code département + Code commune INSEE	String	7	concaténer	renomé en Com_code	concaténation du Code_département et Code_commune_INSEE
Surface_Carrez_premier_lot	Surface Carrez du premier lot	Float	Varchar	Élémentaire	Ne doit pas être nul	NC
Nombre_lots	Nombre de lots	Integer	2	Élémentaire	Ne doit pas être nul	NC
Type_local	Type de local	String	11	Élémentaire	Ne doit pas être nul	NC
Nombre_pieces_principales	Nombre de pièces principales	Integer	2	Élémentaire	Ne doit pas être nul	NC

DICTIONNAIRE DES DONNÉES - Référentiel géographique						
CODE	SIGNIFICATION	TYPE	LONGUEUR	NATURE	REGLE DE GESTION	REGLE DE CALCUL
Com_code	code référant à la commune	String	Variable	Élémentaire	clé primaire	NC
reggrp_nom	Catégorie de la région (Province, Ile de France, DROM-COM)	String	Variable	Élémentaire	valeurs possibles : Province, Ile de France, DROM-COM	NC
reg_nom	Nom de la région	String	Variable	Élémentaire	Ne doit pas être nul	NC
dep_code	Code du département	String	Varchar	Élémentaire	Ne doit pas être nul	NC
dep_nom	nom du département	String	Varchar	Élémentaire	Ne doit pas être nul	NC
COM	nom de la commune	String	Variable	Élémentaire	Ne doit pas être nul	NC

DICTIONNAIRE DES DONNÉES - Données communes						
CODE	SIGNIFICATION	TYPE	LONGUEUR	NATURE	REGLE DE GESTION	REGLE DE CALCUL
COM	Nom de la commune	String	Variable	Élémentaire	Ne doit pas être nul	NC
PTOT	Population totale de la commune	Integer	Variable	Élémentaire	Ne doit pas être nul	NC
Codedep_com	Concat du code dep et com	String	Variable	Élémentaire	Ne doit pas être nul	NC

Dictionnaire des données	Significations
CODE	Nom de la ligne dans le fichier Initial
SIGNIFICATION	Signification de la colonne
TYPE DE VARIABLES	Varchar, Integer, Date, Float, etc.
LONGUEUR	Longueur maximale de la données (pas obligatoire)
NATURE (E/Ca/CO)	Élémentaire, calculer ou concaténer
REGLE DE GESTION	Format de la date (jj/mm/aaaa, jj/mm/aa, etc), ne doit pas être nul, etc.
REGLE DE CALCUL	exemple : Colonne A + Colonne B

- 1 Dictionnaire par fichier
- 1 lexique

Le schéma relationnel normalisé

- 3 tables

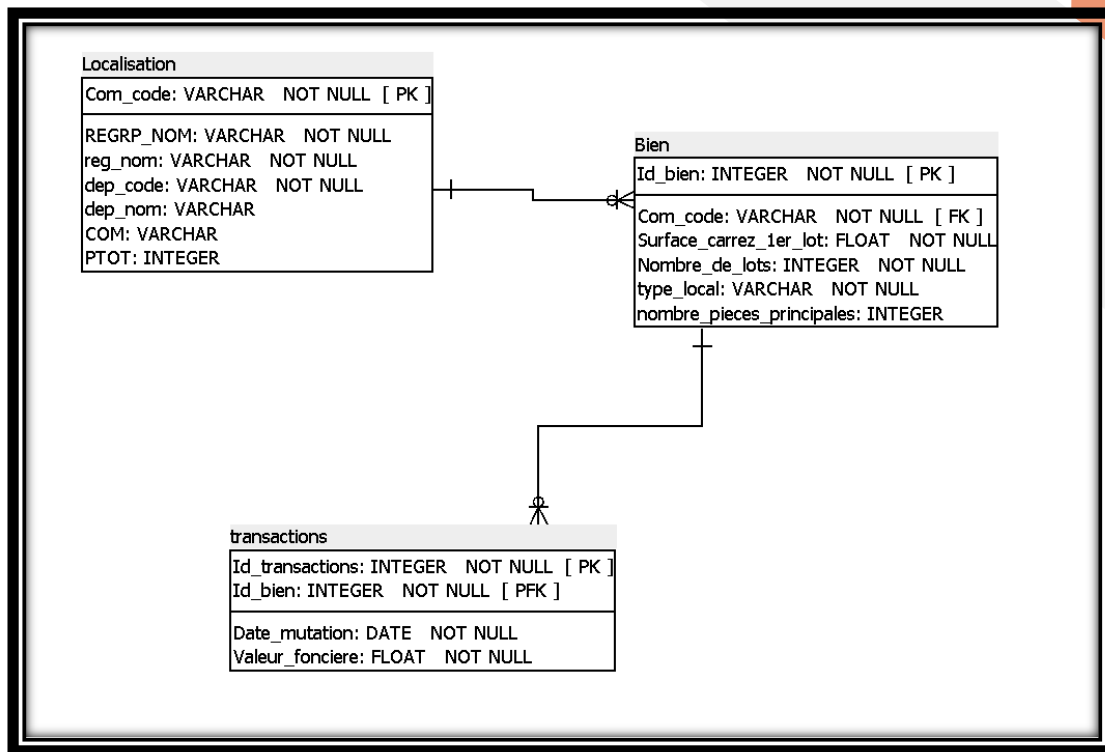
Bien

Localisation

Plusieurs biens peuvent
Être sur un lieu (ex: immeuble)

Transactions

Un bien peut être vendu
Plusieurs fois



La base de données avec les tables créées et les données chargées

	A	B	C	D	E	F
1	id_bien	Com_code	Surface_carrez_1er_lot	Nombre_de_lots	type_local	nombre_pieces_p
4170	34169	972229	75,7	1	Appartement	

< > Bien + : ◀ ▶

- 3 tables

Dont 1 table jointure entre
Ref-geo et données-com

	C	D	E	F	G	
1	reg_nom	dep_code	dep_nom	COM	PTOT	
8917	Corse	2B	Haute-Corse	Chisa		104

< > LOCALISATION + : ◀ ▶

	A	B	C	D	E
1	id_transactions	Id_bien	Date_mutation	Valeur_fonciere	
4170	34011	34011	30/06/2020		

< > transactions + : ◀ ▶

Les requêtes ou screenshot qui permettent de démontrer le bon chargement des données

- Comptage des lignes

De chaque tables

The screenshot shows a database management interface. On the left, a tree view displays the database structure, including tables like 'bien', 'localisation', and 'transactions'. The main window shows a SQL query in the 'Requête' tab:

```
1 SELECT (SELECT COUNT(*) FROM bien) AS bien,  
2        (SELECT COUNT(*) FROM localisation) AS localisation,  
3        (SELECT COUNT(*) FROM transactions) AS transactions;
```

Below the query, the 'Table' tab shows the results of the query, which are the row counts for each table:

	bien	localisation	transactions
1	34169	38916	34169

The interface also includes a toolbar with various icons for database operations and a status bar at the bottom indicating 'Nombre de lignes chargées : 1'.



Requêtes SQL et résultats

Interprétation des résultats

Explication des commandes

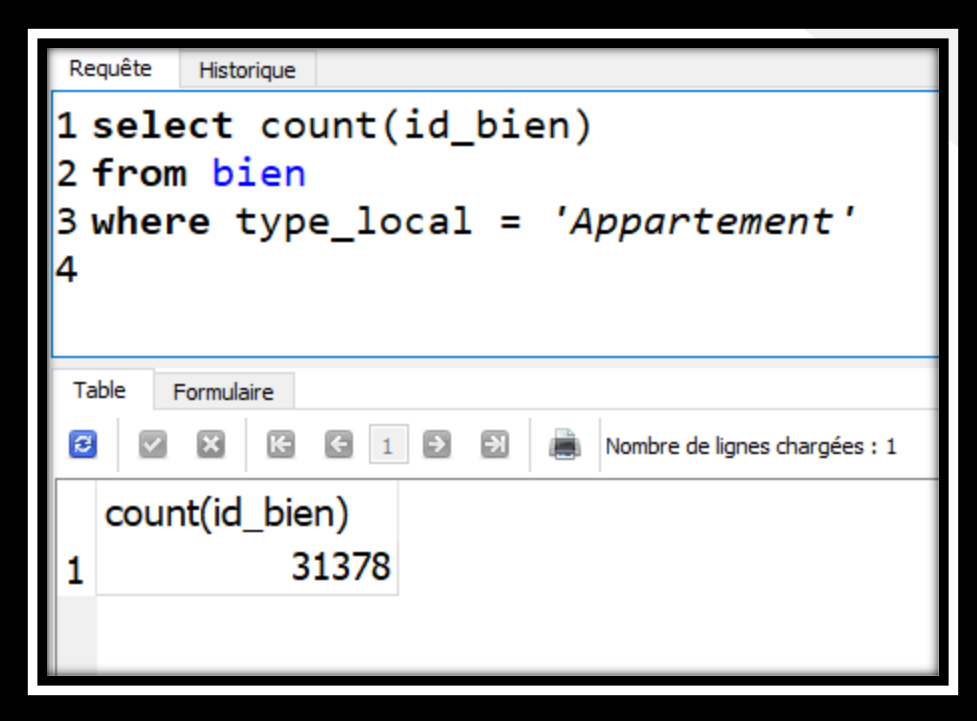
Requête 1

1. Nombre total d'appartements vendus au 1er semestre 2020.

On peut ajouter si les données étaient plus larges :

AND SUBSTR(t.Date_mutation, 7) is '2020'

AND SUBSTR(t.Date_mutation, 4, 2) BETWEEN
'01' AND '06'



The screenshot shows a database query interface with two tabs: 'Requête' (selected) and 'Historique'. The SQL query is displayed in the 'Requête' tab:

```
1 select count(id_bien)
2 from bien
3 where type_local = 'Appartement'
4
```

Below the query, there are two tabs: 'Table' (selected) and 'Formulaire'. The 'Table' tab displays the results of the query in a table format. The table has one column, 'count(id_bien)', and one row with the value 31378. The interface also includes a toolbar with various icons (refresh, check, close, back, forward, etc.) and a status bar indicating 'Nombre de lignes chargées : 1'.

	count(id_bien)
1	31378

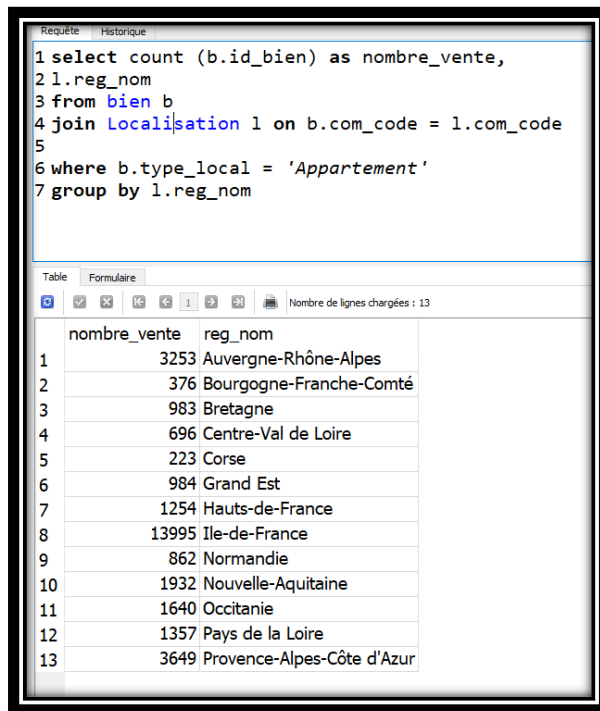
Requête 2

2. Le nombre de ventes d'appartement par région pour le 1er semestre 2020.

On peut ajouter si les données étaient plus larges :

AND SUBSTR(t.Date_mutation, 7) is '2020'

AND SUBSTR(t.Date_mutation, 4, 2) BETWEEN
'01' AND '06'

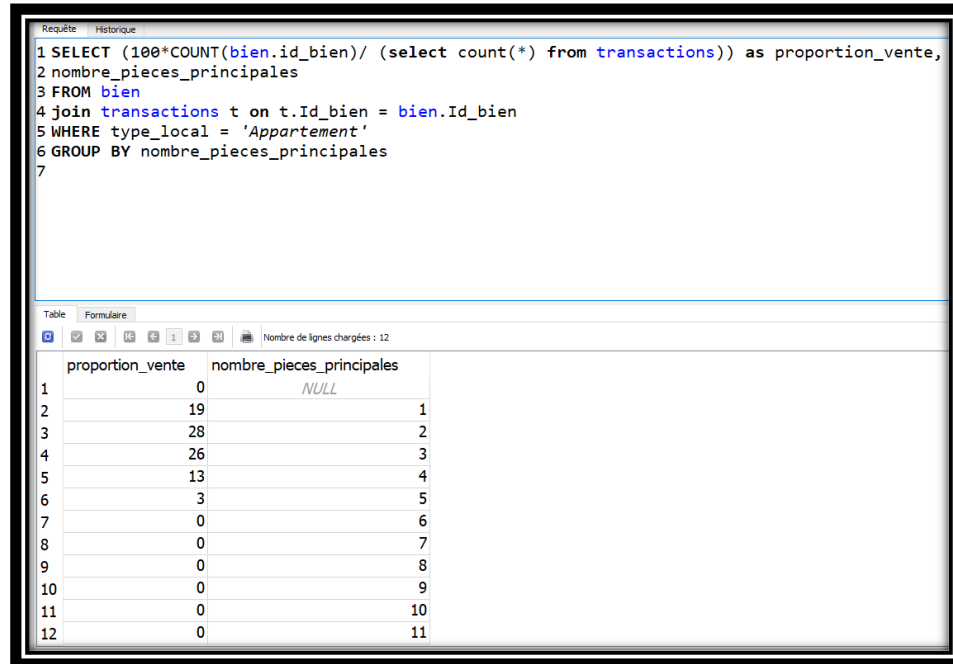


The screenshot shows a database query tool interface. The top pane displays a SQL query: `1 select count (b.id_bien) as nombre_vente, 2 l.reg_nom 3 from bien b 4 join Localisation l on b.com_code = l.com_code 5 6 where b.type_local = 'Appartement' 7 group by l.reg_nom`. The bottom pane shows the results in a table with two columns: `nombre_vente` and `reg_nom`. The results are listed in 13 rows, numbered 1 to 13 on the left. The data represents the number of apartment sales per region for the first semester of 2020.

	nombre_vente	reg_nom
1	3253	Auvergne-Rhône-Alpes
2	376	Bourgogne-Franche-Comté
3	983	Bretagne
4	696	Centre-Val de Loire
5	223	Corse
6	984	Grand Est
7	1254	Hauts-de-France
8	13995	Ile-de-France
9	862	Normandie
10	1932	Nouvelle-Aquitaine
11	1640	Occitanie
12	1357	Pays de la Loire
13	3649	Provence-Alpes-Côte d'Azur

Requête 3

3. Proportion des ventes d'appartements par le nombre de pièces.



The screenshot shows a database query interface with a 'Requête' tab. The SQL query is as follows:

```
1 SELECT (100*COUNT(bien.id_bien)/ (select count(*) from transactions)) as proportion_vente,  
2 nombre_pieces_principales  
3 FROM bien  
4 join transactions t on t.Id_bien = bien.Id_bien  
5 WHERE type_local = 'Appartement'  
6 GROUP BY nombre_pieces_principales  
7
```

Below the query, there is a 'Table' tab showing the results. The interface includes a toolbar with icons for various actions and a status bar indicating 'Nombre de lignes chargées : 12'.

	proportion_vente	nombre_pieces_principales
1	0	NULL
2	19	1
3	28	2
4	26	3
5	13	4
6	3	5
7	0	6
8	0	7
9	0	8
10	0	9
11	0	10
12	0	11

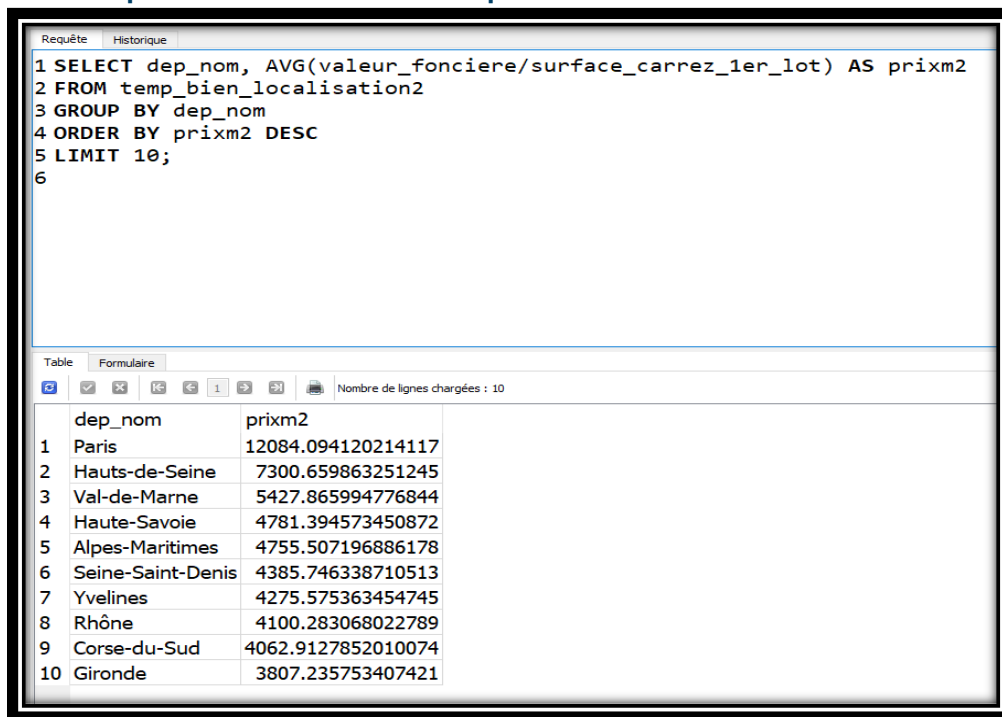
Requête speciale

Creation d'une table temporaire

```
2 CREATE TEMPORARY TABLE temp_bien_localisation2 AS
3 SELECT
4     transactions.*,
5     bien.*,
6     localisation.*
7 FROM
8     transactions
9 JOIN
10    bien ON transactions.id_bien = bien.id_bien
11 JOIN |
12    localisation ON bien.Com_code = localisation.Com_code;
13
```

Requête 4

4. Liste des 10 départements où le prix du mètre carré est le plus élevé.



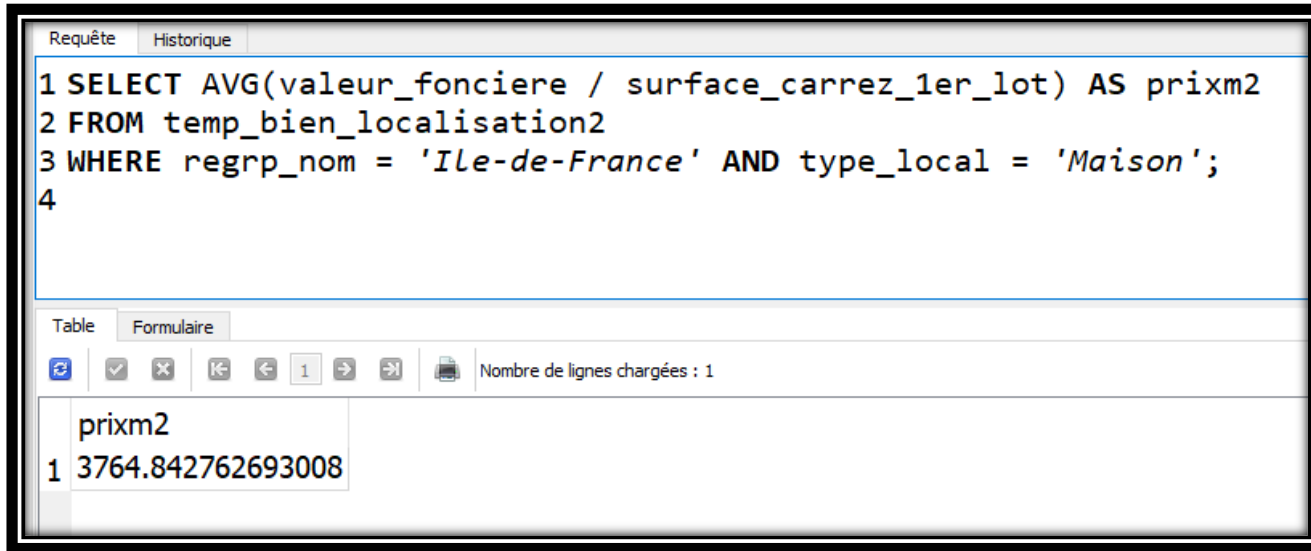
The screenshot shows a database query interface with two tabs: 'Requête' and 'Historique'. The 'Requête' tab is active, displaying a SQL query. Below the query, there is a toolbar with icons for various database operations. The results are displayed in a table with two columns: 'dep_nom' and 'prixm2'. The table lists the top 10 departments by average price per square meter.

```
1 SELECT dep_nom, AVG(valeur_fonciere/surface_carrez_1er_lot) AS prixm2
2 FROM temp_bien_localisation2
3 GROUP BY dep_nom
4 ORDER BY prixm2 DESC
5 LIMIT 10;
6
```

	dep_nom	prixm2
1	Paris	12084.094120214117
2	Hauts-de-Seine	7300.659863251245
3	Val-de-Marne	5427.865994776844
4	Haute-Savoie	4781.394573450872
5	Alpes-Maritimes	4755.507196886178
6	Seine-Saint-Denis	4385.746338710513
7	Yvelines	4275.575363454745
8	Rhône	4100.283068022789
9	Corse-du-Sud	4062.9127852010074
10	Gironde	3807.235753407421

Requête 5

5. Prix moyen du mètre carré d'une maison en Île-de-France.



The screenshot shows a database query interface with two tabs: 'Requête' (selected) and 'Historique'. The query text is as follows:

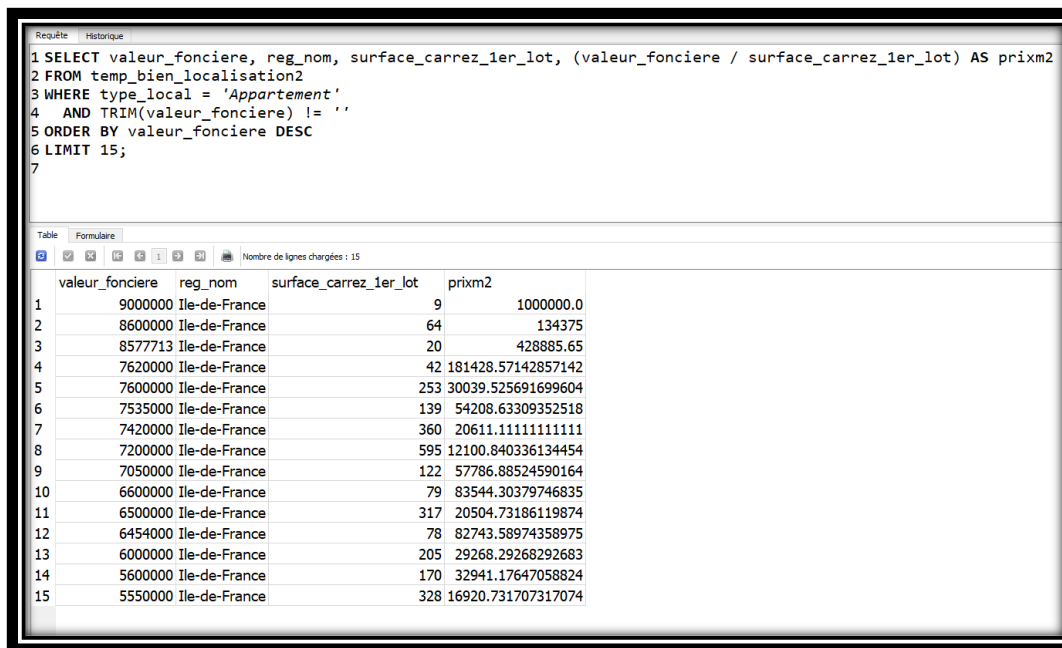
```
1 SELECT AVG(valeur_fonciere / surface_carrez_1er_lot) AS prixm2
2 FROM temp_bien_localisation2
3 WHERE regroup_nom = 'Ile-de-France' AND type_local = 'Maison';
4
```

Below the query editor, there is a 'Table' tab and a 'Formulaire' tab. The 'Table' tab is active, displaying a table with one row and one column. The column header is 'prixm2' and the value is '3764.842762693008'. To the right of the table, it says 'Nombre de lignes chargées : 1'.

	prixm2
1	3764.842762693008

Requête 6

6. Liste des 10 appartements les plus chers avec la région et le nombre de mètres carrés.



The screenshot shows a database query interface with a SQL query editor at the top and a results table below. The query is a SELECT statement that filters for 'Appartement' type, excludes empty values, and orders by 'valeur_fonciere' in descending order, limiting the results to 15 rows. The results table displays the top 15 most expensive apartments, including their 'valeur_fonciere', 'reg_nom' (all 'Ile-de-France'), 'surface_carrez_1er_lot', and a calculated 'prixm2' value.

```
1 SELECT valeur_fonciere, reg_nom, surface_carrez_1er_lot, (valeur_fonciere / surface_carrez_1er_lot) AS prixm2
2 FROM temp_bien_localisation2
3 WHERE type_local = 'Appartement'
4 AND TRIM(valeur_fonciere) != ''
5 ORDER BY valeur_fonciere DESC
6 LIMIT 15;
7
```

	valeur_fonciere	reg_nom	surface_carrez_1er_lot	prixm2
1	9000000	Ile-de-France	9	1000000.0
2	8600000	Ile-de-France	64	134375
3	8577713	Ile-de-France	20	428885.65
4	7620000	Ile-de-France	42	181428.57142857142
5	7600000	Ile-de-France	253	30039.525691699604
6	7535000	Ile-de-France	139	54208.63309352518
7	7420000	Ile-de-France	360	20611.11111111111
8	7200000	Ile-de-France	595	12100.840336134454
9	7050000	Ile-de-France	122	57786.88524590164
10	6600000	Ile-de-France	79	83544.30379746835
11	6500000	Ile-de-France	317	20504.73186119874
12	6454000	Ile-de-France	78	82743.58974358975
13	6000000	Ile-de-France	205	29268.29268292683
14	5600000	Ile-de-France	170	32941.17647058824
15	5550000	Ile-de-France	328	16920.731707317074

Requête 7

7. Taux d'évolution du nombre de ventes entre le premier et le second Trimestre de 2020.

Requête

Historique

```
1 WITH ventestri1 AS (  
2 SELECT COUNT(*) AS ventestri1  
3 FROM temp_bien_localisation2  
4 WHERE SUBSTR(date_mutation, 4, 2) IN ('01', '02', '03')  
5 ),  
6 ventestri2 AS (  
7 SELECT COUNT(*) AS ventestri2  
8 FROM temp_bien_localisation2  
9 WHERE SUBSTR(date_mutation, 4, 2) IN ('04', '05', '06')  
10 )  
11 SELECT ventestri1, ventestri2 , (((ventestri2-ventestri1)*100)/ventestri1) as évolution_pourcent  
12 FROM ventestri1, ventestri2  
13
```

Table

Formulaire

✓

✗

🔍

⏪

1

⏩

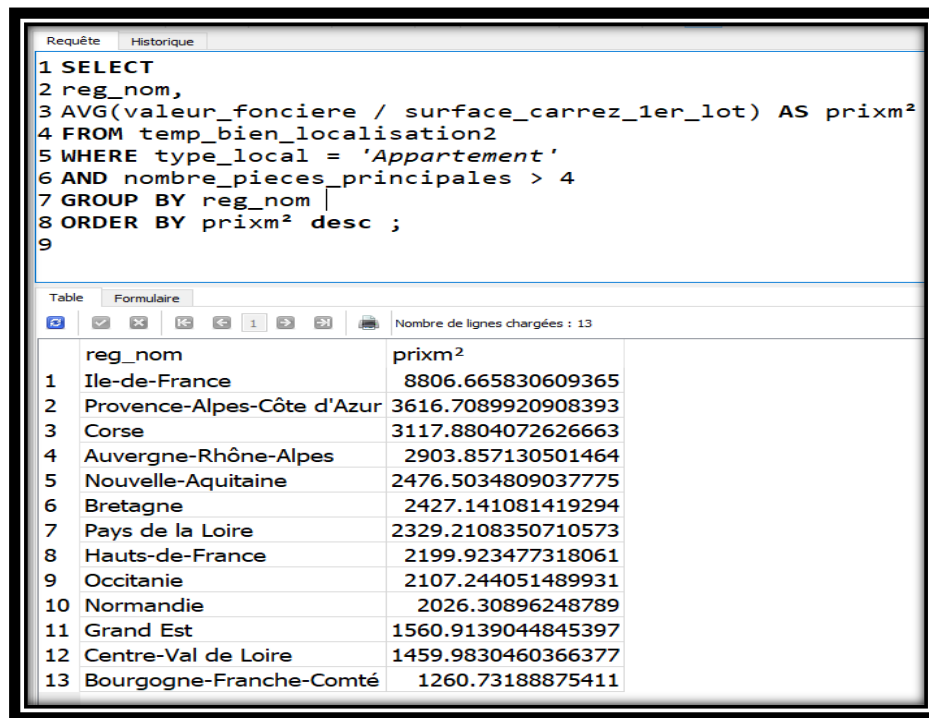
🖨

Nombre de lignes chargées : 1

	ventestri1	ventestri2	évolution_pourcent
1	16688	17286	3

Requête 8

8. Le classement des régions par rapport au prix au mètre carré des Appartements de plus de 4 pièces












The screenshot shows a database query tool interface. At the top, there are tabs for 'Requête' and 'Historique'. The 'Requête' tab is active, displaying a SQL query. Below the query, there are tabs for 'Table' and 'Formulaire'. The 'Table' tab is active, showing a table with 13 rows and 2 columns. The table is titled 'Nombre de lignes chargées : 13'. The columns are 'reg_nom' and 'prixm²'. The rows are numbered 1 to 13, corresponding to the regions listed in the table.

```
1 SELECT
2 reg_nom,
3 AVG(valeur_fonciere / surface_carrez_1er_lot) AS prixm²
4 FROM temp_bien_localisation2
5 WHERE type_local = 'Appartement'
6 AND nombre_pieces_principales > 4
7 GROUP BY reg_nom
8 ORDER BY prixm² desc ;
9
```

	reg_nom	prixm²
1	Ile-de-France	8806.665830609365
2	Provence-Alpes-Côte d'Azur	3616.7089920908393
3	Corse	3117.8804072626663
4	Auvergne-Rhône-Alpes	2903.857130501464
5	Nouvelle-Aquitaine	2476.5034809037775
6	Bretagne	2427.141081419294
7	Pays de la Loire	2329.2108350710573
8	Hauts-de-France	2199.923477318061
9	Occitanie	2107.244051489931
10	Normandie	2026.30896248789
11	Grand Est	1560.9139044845397
12	Centre-Val de Loire	1459.9830460366377
13	Bourgogne-Franche-Comté	1260.73188875411

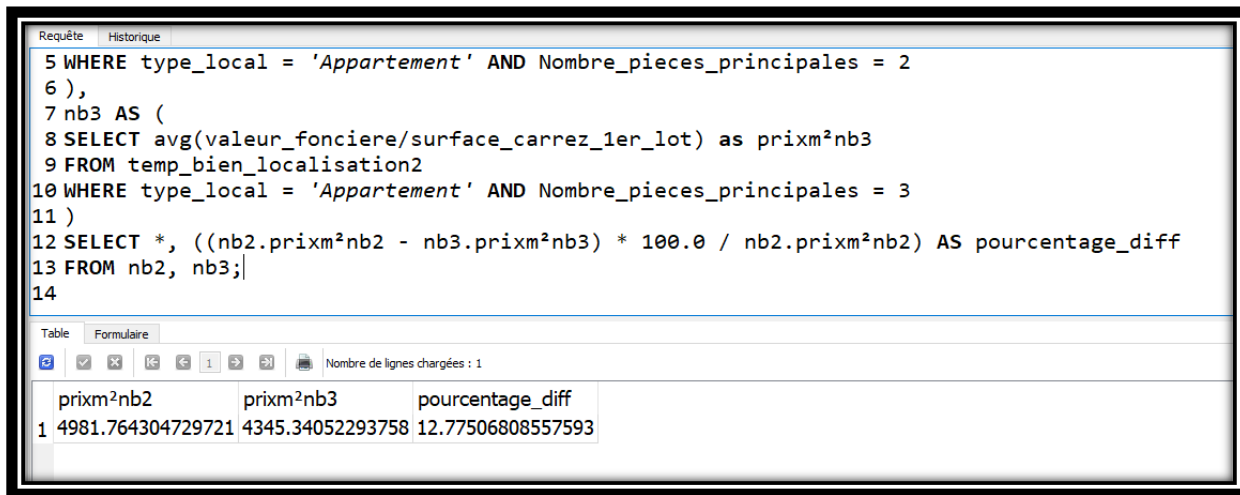
Requête 9

9. Liste des communes ayant eu au moins 50 ventes au 1er trimestre

Requête		Historique
<pre>1 SELECT COM, COUNT(id_transactions) AS nombre_transactions 2 FROM temp_bien_localisation2 3 WHERE SUBSTR(date_mutation, 4, 2) IN ('01', '02', '03') 4 GROUP BY COM 5 HAVING COUNT(id_transactions) > 50 6 order by nombre_transactions 7 </pre>		
Table		Formulaire
        		Nombre de lignes chargées : 47
COM	nombre_transactions	
1 Puteaux	53	
2 Ajaccio	54	
3 Versailles	54	
4 Saint-Maur-des-Fossés	56	
5 Levallois-Perret	59	
6 Toulon	59	
7 Paris 4e Arrondissement	60	
8 Paris 2e Arrondissement	61	
9 Rennes	61	
10 La Ciotat	62	
11 Paris 8e Arrondissement	62	
12 Sète	62	
13 Nîmes	63	
14 Angers	64	
15 Montreuil	65	

Requête 10

10. Différence en pourcentage du prix au mètre carré entre un Appartement de 2 pièces et un appartement de 3 pièces.



The screenshot shows a SQL query editor with a query window and a results table. The query is as follows:

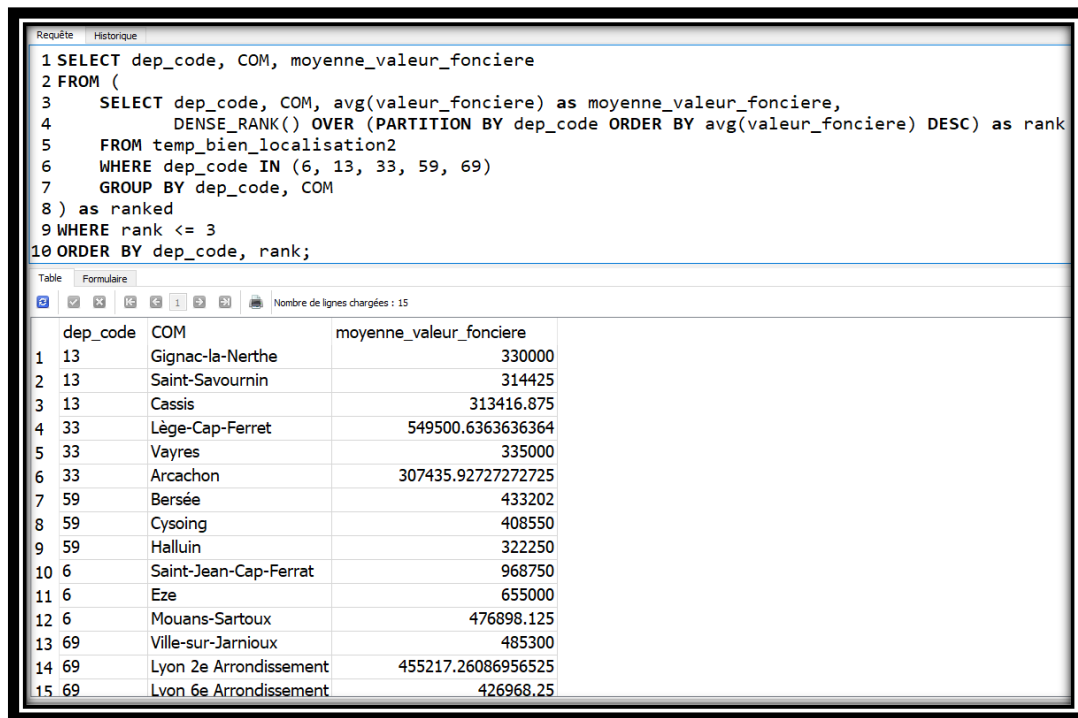
```
5 WHERE type_local = 'Appartement' AND Nombre_pieces_principales = 2
6 ),
7 nb3 AS (
8 SELECT avg(valeur_fonciere/surface_carrez_1er_lot) as prixm²nb3
9 FROM temp_bien_localisation2
10 WHERE type_local = 'Appartement' AND Nombre_pieces_principales = 3
11 )
12 SELECT *, ((nb2.prixm²nb2 - nb3.prixm²nb3) * 100.0 / nb2.prixm²nb2) AS pourcentage_diff
13 FROM nb2, nb3;
14
```

Below the query editor, there is a results table with the following columns: prixm²nb2, prixm²nb3, and pourcentage_diff. The table contains one row of data.

	prixm²nb2	prixm²nb3	pourcentage_diff
1	4981.764304729721	4345.34052293758	12.77506808557593

Requête 11

11. Les moyennes de valeurs foncières pour le top 3 des communes des Départements 6, 13, 33, 59 et 69.



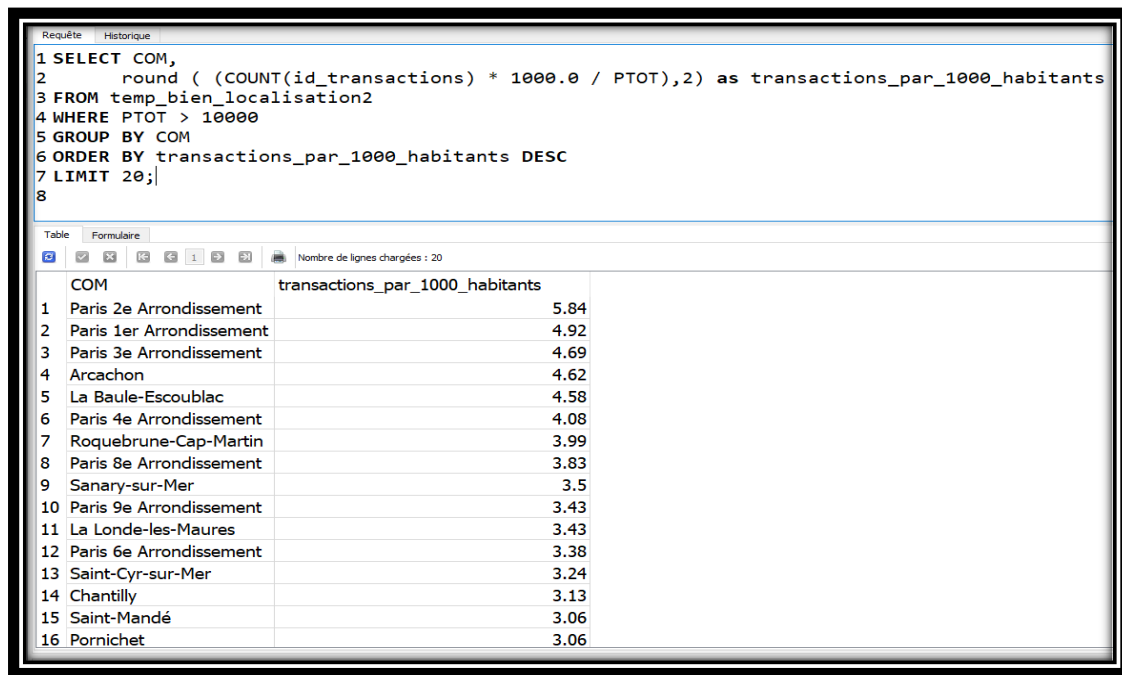
The screenshot shows a database query interface with a SQL query editor and a results table. The query is a window function that ranks communes by their average land value within specific departments. The results table displays the top 3 communes for each department, ordered by their rank.

```
1 SELECT dep_code, COM, moyenne_valeur_fonciere
2 FROM (
3     SELECT dep_code, COM, avg(valeur_fonciere) as moyenne_valeur_fonciere,
4         DENSE_RANK() OVER (PARTITION BY dep_code ORDER BY avg(valeur_fonciere) DESC) as rank
5     FROM temp_bien_localisation2
6     WHERE dep_code IN (6, 13, 33, 59, 69)
7     GROUP BY dep_code, COM
8 ) as ranked
9 WHERE rank <= 3
10 ORDER BY dep_code, rank;
```

	dep_code	COM	moyenne_valeur_fonciere
1	13	Gignac-la-Nerthe	330000
2	13	Saint-Savournin	314425
3	13	Cassis	313416.875
4	33	Lège-Cap-Ferret	549500.6363636364
5	33	Vayres	335000
6	33	Arcachon	307435.92727272725
7	59	Bersée	433202
8	59	Cysoing	408550
9	59	Halluin	322250
10	6	Saint-Jean-Cap-Ferrat	968750
11	6	Eze	655000
12	6	Mouans-Sartoux	476898.125
13	69	Ville-sur-Jarnioux	485300
14	69	Lyon 2e Arrondissement	455217.26086956525
15	69	Lyon 6e Arrondissement	426968.25

Requête 12

12. Les 20 communes avec le plus de transactions pour 1000 habitants pour les communes qui dépassent les 10 000 habitants.



The screenshot shows a database query interface. At the top, there are tabs for 'Requête' and 'Historique'. Below them is a text area containing a SQL query. The query is as follows:

```
1 SELECT COM,  
2     round ( (COUNT(id_transactions) * 1000.0 / PTOT),2) as transactions_par_1000_habitants  
3 FROM temp_bien_localisation2  
4 WHERE PTOT > 10000  
5 GROUP BY COM  
6 ORDER BY transactions_par_1000_habitants DESC  
7 LIMIT 20;  
8
```

Below the query text area, there are tabs for 'Table' and 'Formulaire'. The 'Table' tab is selected, and it shows a table with 2 columns: 'COM' and 'transactions_par_1000_habitants'. The table contains 16 rows of data, numbered 1 to 16. The 'Nombre de lignes chargées : 20' is displayed at the top right of the table area.

	COM	transactions_par_1000_habitants
1	Paris 2e Arrondissement	5.84
2	Paris 1er Arrondissement	4.92
3	Paris 3e Arrondissement	4.69
4	Arcachon	4.62
5	La Baule-Escoublac	4.58
6	Paris 4e Arrondissement	4.08
7	Roquebrune-Cap-Martin	3.99
8	Paris 8e Arrondissement	3.83
9	Sanary-sur-Mer	3.5
10	Paris 9e Arrondissement	3.43
11	La Londe-les-Maures	3.43
12	Paris 6e Arrondissement	3.38
13	Saint-Cyr-sur-Mer	3.24
14	Chantilly	3.13
15	Saint-Mandé	3.06
16	Pornichet	3.06



Conclusion

Remerciement