

# DATA Immo

Requêtes - MySQL

Équipe DATA  
Janvier 2023



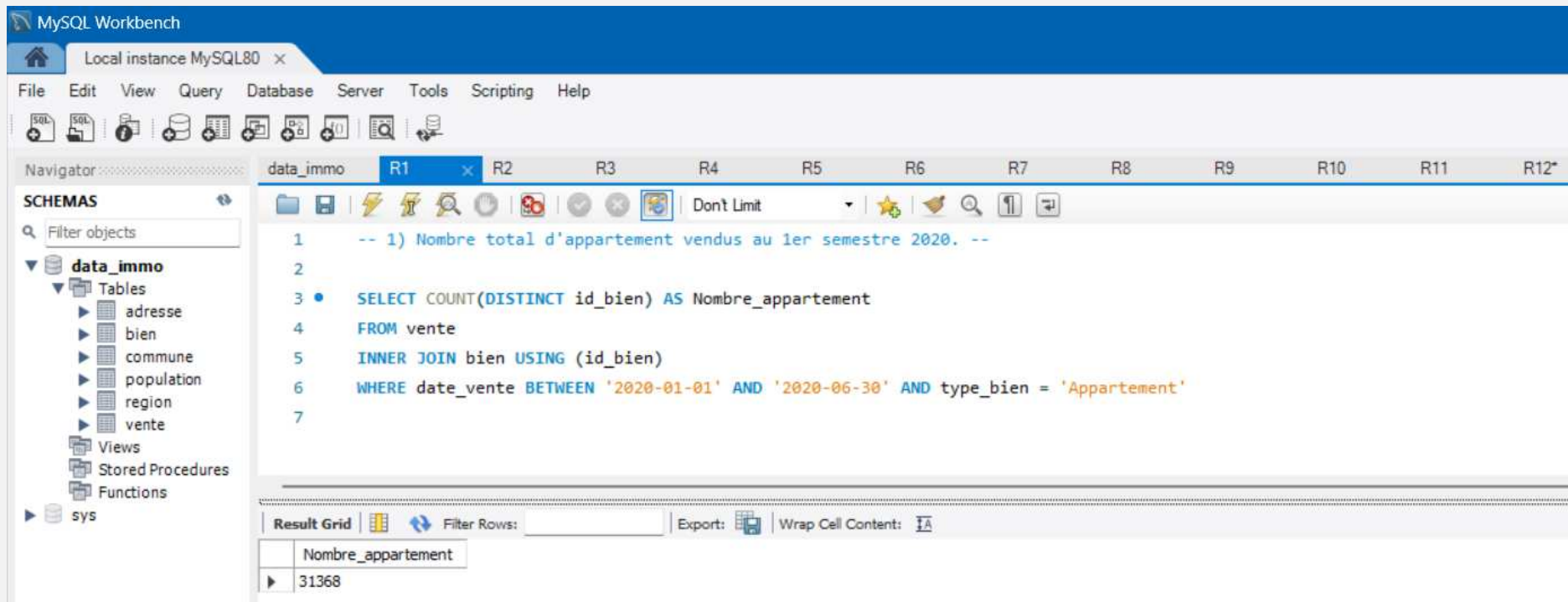
# Sommaire

## 1. Code MySQL – 12 requêtes



# Code MySQL – 12 requêtes

## 1) Nombre total d'appartement vendus au 1er semestre 2020



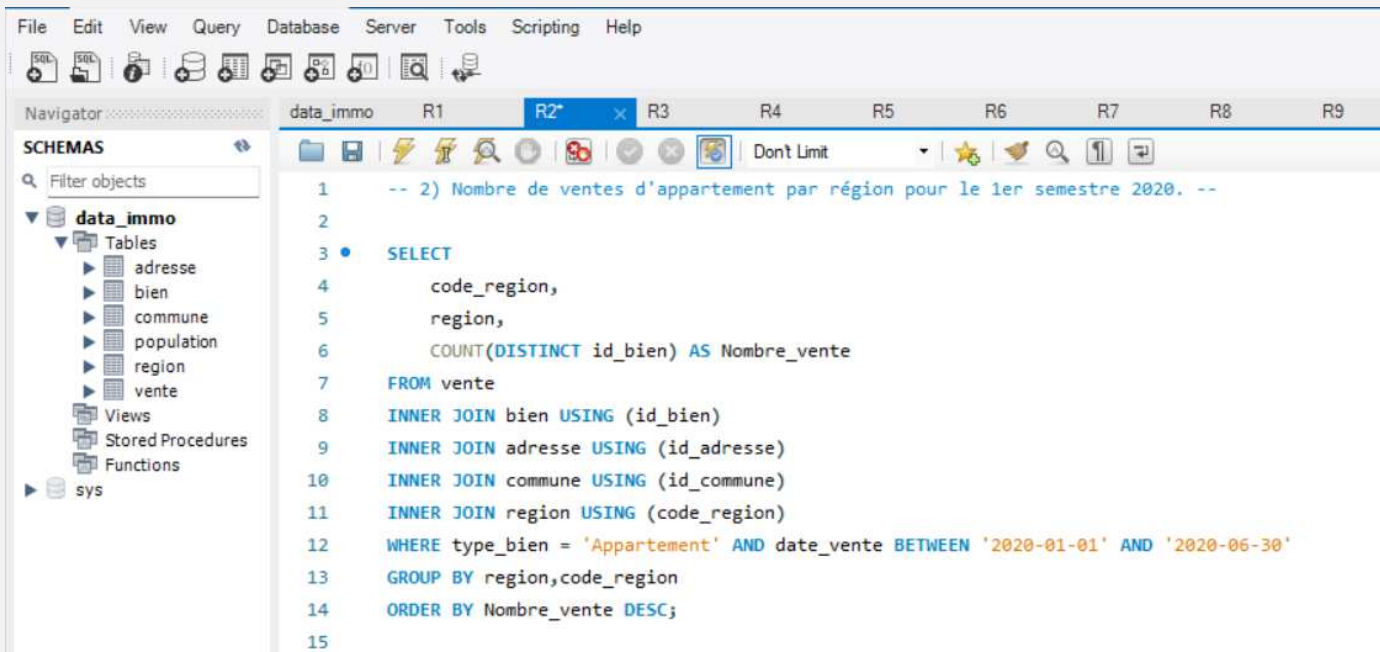
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'data\_immo' selected, showing tables like 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The main editor window shows a SQL query in the 'R1' tab:

```
1  -- 1) Nombre total d'appartement vendus au 1er semestre 2020. --
2
3  •  SELECT COUNT(DISTINCT id_bien) AS Nombre_appartement
4     FROM vente
5     INNER JOIN bien USING (id_bien)
6     WHERE date_vente BETWEEN '2020-01-01' AND '2020-06-30' AND type_bien = 'Appartement'
7
```

The bottom panel shows the 'Result Grid' with the following data:

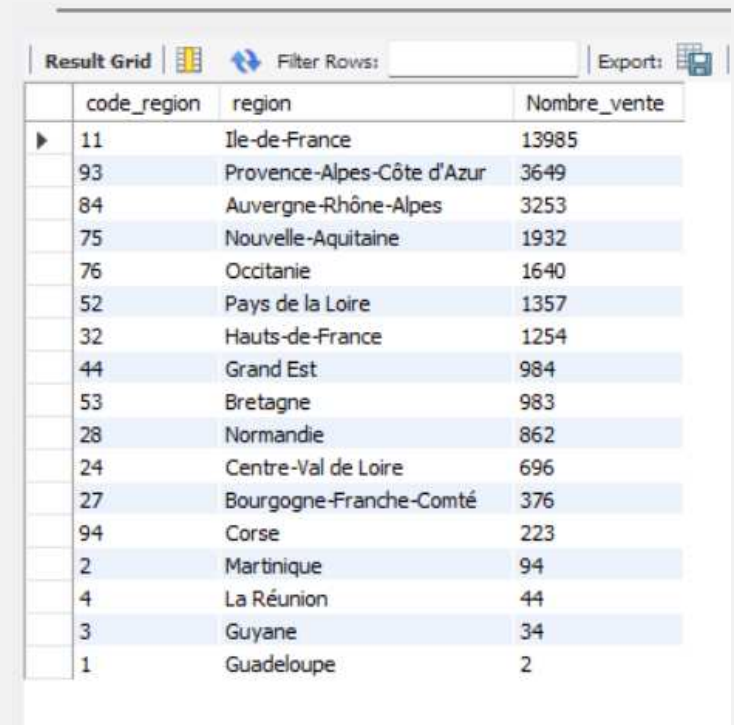
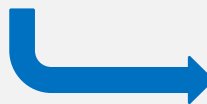
Nombre_appartement
31368

## 2) Nombre de ventes d'appartement par région pour le 1er semestre 2020.



The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar. The left sidebar displays a 'SCHEMAS' tree with 'data\_immo' expanded, showing tables like 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The main editor window shows a SQL query in a tab labeled 'R2\*'. The query is as follows:

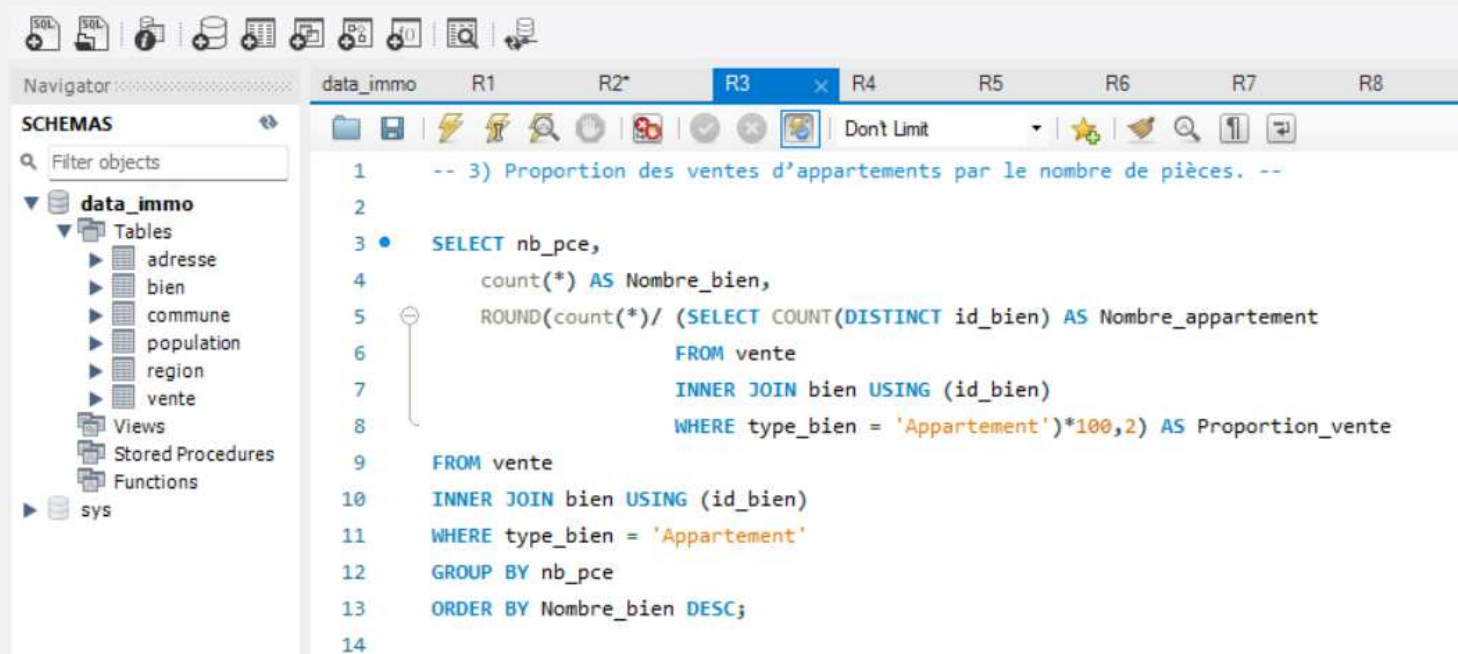
```
1  -- 2) Nombre de ventes d'appartement par région pour le 1er semestre 2020. --
2
3  •  SELECT
4      code_region,
5      region,
6      COUNT(DISTINCT id_bien) AS Nombre_vente
7  FROM vente
8  INNER JOIN bien USING (id_bien)
9  INNER JOIN adresse USING (id_adresse)
10 INNER JOIN commune USING (id_commune)
11 INNER JOIN region USING (code_region)
12 WHERE type_bien = 'Appartement' AND date_vente BETWEEN '2020-01-01' AND '2020-06-30'
13 GROUP BY region,code_region
14 ORDER BY Nombre_vente DESC;
15
```



The screenshot shows the 'Result Grid' of the SQL IDE, displaying the results of the query. The grid has columns for 'code\_region', 'region', and 'Nombre\_vente'. The results are ordered by 'Nombre\_vente' in descending order.

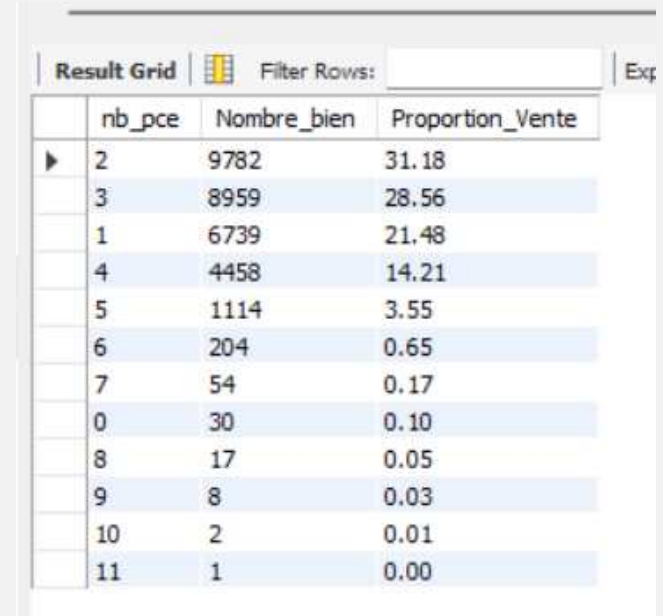
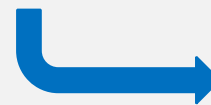
	code_region	region	Nombre_vente
▶	11	Ile-de-France	13985
	93	Provence-Alpes-Côte d'Azur	3649
	84	Auvergne-Rhône-Alpes	3253
	75	Nouvelle-Aquitaine	1932
	76	Occitanie	1640
	52	Pays de la Loire	1357
	32	Hauts-de-France	1254
	44	Grand Est	984
	53	Bretagne	983
	28	Normandie	862
	24	Centre-Val de Loire	696
	27	Bourgogne-Franche-Comté	376
	94	Corse	223
	2	Martinique	94
	4	La Réunion	44
	3	Guyane	34
	1	Guadeloupe	2

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



The screenshot shows a SQL IDE interface with a sidebar on the left displaying a database schema named 'data\_immo'. The schema includes tables: 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The main editor window shows a SQL query in a tab labeled 'R3'. The query is as follows:

```
1  -- 3) Proportion des ventes d'appartements par le nombre de pièces. --
2
3  SELECT nb_pce,
4         count(*) AS Nombre_bien,
5         ROUND(count(*)/ (SELECT COUNT(DISTINCT id_bien) AS Nombre_appartement
6                           FROM vente
7                           INNER JOIN bien USING (id_bien)
8                           WHERE type_bien = 'Appartement')*100,2) AS Proportion_vente
9
10 FROM vente
11 INNER JOIN bien USING (id_bien)
12 WHERE type_bien = 'Appartement'
13 GROUP BY nb_pce
14 ORDER BY Nombre_bien DESC;
```



The screenshot shows the 'Result Grid' of the SQL IDE. It displays the results of the query, showing the number of rooms (nb\_pce), the number of apartments (Nombre\_bien), and the proportion of sales (Proportion\_Vente) for each number of rooms. The results are ordered by the number of apartments in descending order.

	nb_pce	Nombre_bien	Proportion_Vente
▶	2	9782	31.18
	3	8959	28.56
	1	6739	21.48
	4	4458	14.21
	5	1114	3.55
	6	204	0.65
	7	54	0.17
	0	30	0.10
	8	17	0.05
	9	8	0.03
	10	2	0.01
	11	1	0.00

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

Navigator

data\_immo R1 R2\* R3 R4 R5 R6 R7 R8 R9 R10

SCHEMAS

Filter objects

data\_immo

- Tables
  - adresse
  - bien
  - commune
  - population
  - region
  - vente
- Views
- Stored Procedures
- Functions

sys

```
1 -- 4) Liste des 10 départements où le prix du mètre carré est le plus élevé. --
2
3 SELECT no_dept, departement, ROUND(avg(valeur_fonciere/surface_carrez_lot_un),2) AS prix_m²
4 FROM vente
5 INNER JOIN bien USING (id_bien)
6 INNER JOIN adresse USING (id_adresse)
7 INNER JOIN commune USING (id_commune)
8 WHERE surface_carrez_lot_un != 0
9 GROUP BY no_dept, departement
10 ORDER BY prix_m² DESC
11 LIMIT 10
```



Result Grid | Filter Rows:

	no_dept	departement	prix_m²
▶	75	Paris	12045.77
	92	Hauts-de-Seine	7219.99
	94	Val-de-Marne	5340.51
	6	Alpes-Maritimes	4696.83
	74	Haute-Savoie	4667.13
	93	Seine-Saint-Denis	4320.57
	78	Yvelines	4225.25
	69	Rhône	4059.31
	2A	Corse-du-Sud	4010.6
	33	Gironde	3764.14



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

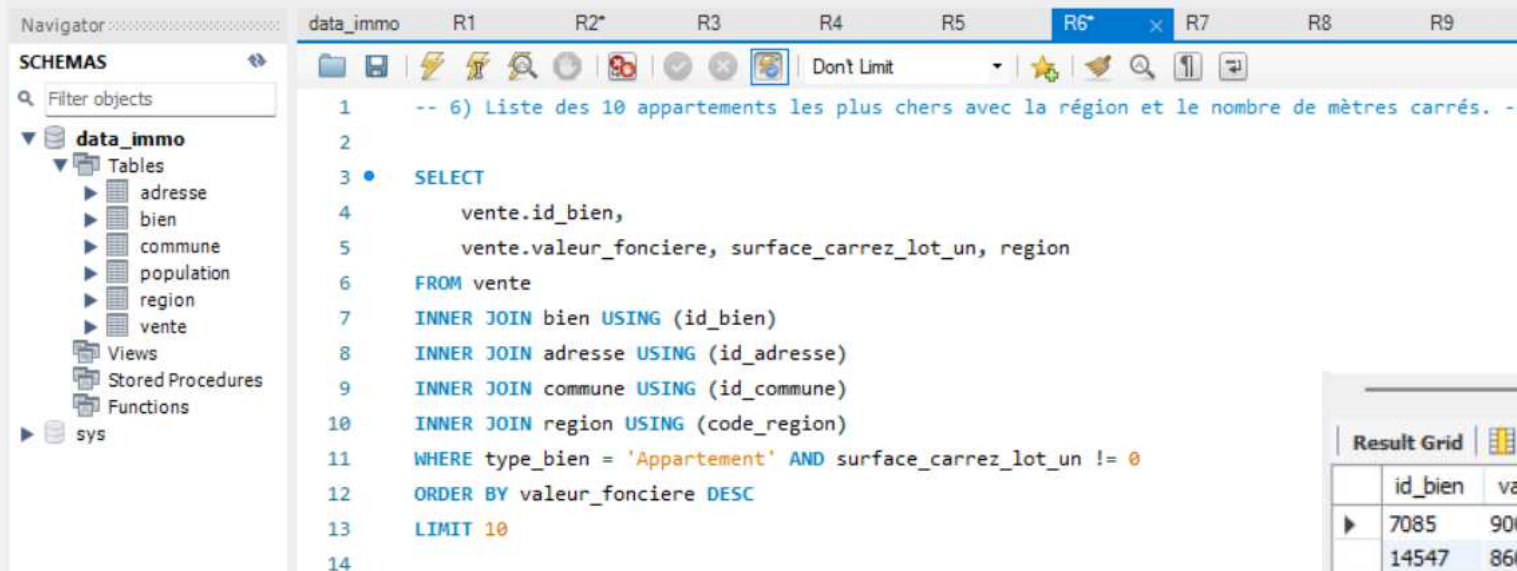
The screenshot shows a SQL IDE interface with a Navigator on the left and a query editor on the right. The Navigator displays a schema named 'data\_immo' with several tables: 'adresse', 'bien', 'commune', 'population', 'region', 'vente', 'Views', 'Stored Procedures', and 'Functions'. The 'commune' table is selected. The query editor shows a SQL query to calculate the average price per square meter of houses in Île-de-France. The query is as follows:

```
-- 5) Prix moyen du mètre carré d'une maison en Île-de-France. --  
  
SELECT ROUND(avg(valeur_fonciere/surface_carrez_lot_un),2) AS prix_m²_moyen_IdF  
FROM vente  
INNER JOIN bien USING (id_bien)  
INNER JOIN adresse USING (id_adresse)  
INNER JOIN commune USING (id_commune)  
INNER JOIN region USING (code_region)  
WHERE type_bien = 'Maison' AND type_region = 'Ile-de-France' AND surface_carrez_lot_un != 0
```

The result grid at the bottom shows the output of the query:

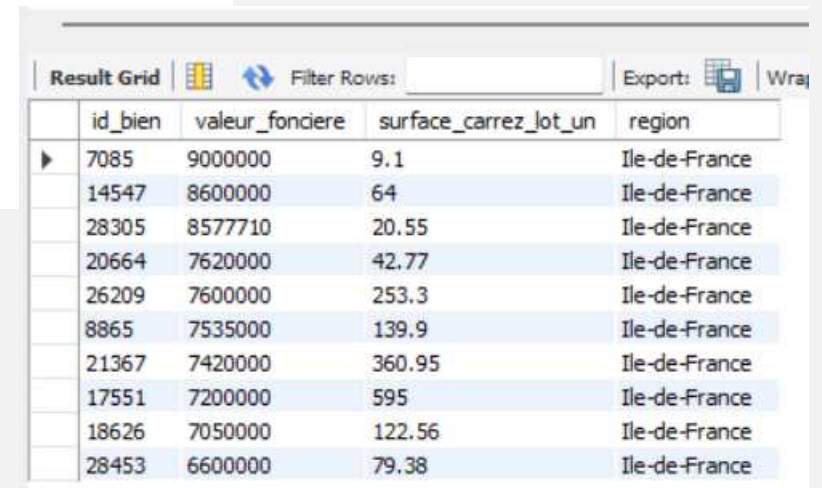
	prix_m²_moyen_IdF
▶	3745.01

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



The screenshot shows a SQL IDE interface. On the left is a 'SCHEMAS' panel with a tree view showing the 'data\_immo' database containing tables: 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The main editor window shows a SQL query with line numbers 1 through 14. The query is as follows:

```
1  -- 6) Liste des 10 appartements les plus chers avec la région et le nombre de mètres carrés. --
2
3  SELECT
4      vente.id_bien,
5      vente.valeur_fonciere, surface_carrez_lot_un, region
6  FROM vente
7  INNER JOIN bien USING (id_bien)
8  INNER JOIN adresse USING (id_adresse)
9  INNER JOIN commune USING (id_commune)
10 INNER JOIN region USING (code_region)
11 WHERE type_bien = 'Appartement' AND surface_carrez_lot_un != 0
12 ORDER BY valeur_fonciere DESC
13 LIMIT 10
14
```



The screenshot shows a 'Result Grid' with the following data:

	id_bien	valeur_fonciere	surface_carrez_lot_un	region
▶	7085	9000000	9.1	Ile-de-France
	14547	8600000	64	Ile-de-France
	28305	8577710	20.55	Ile-de-France
	20664	7620000	42.77	Ile-de-France
	26209	7600000	253.3	Ile-de-France
	8865	7535000	139.9	Ile-de-France
	21367	7420000	360.95	Ile-de-France
	17551	7200000	595	Ile-de-France
	18626	7050000	122.56	Ile-de-France
	28453	6600000	79.38	Ile-de-France



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

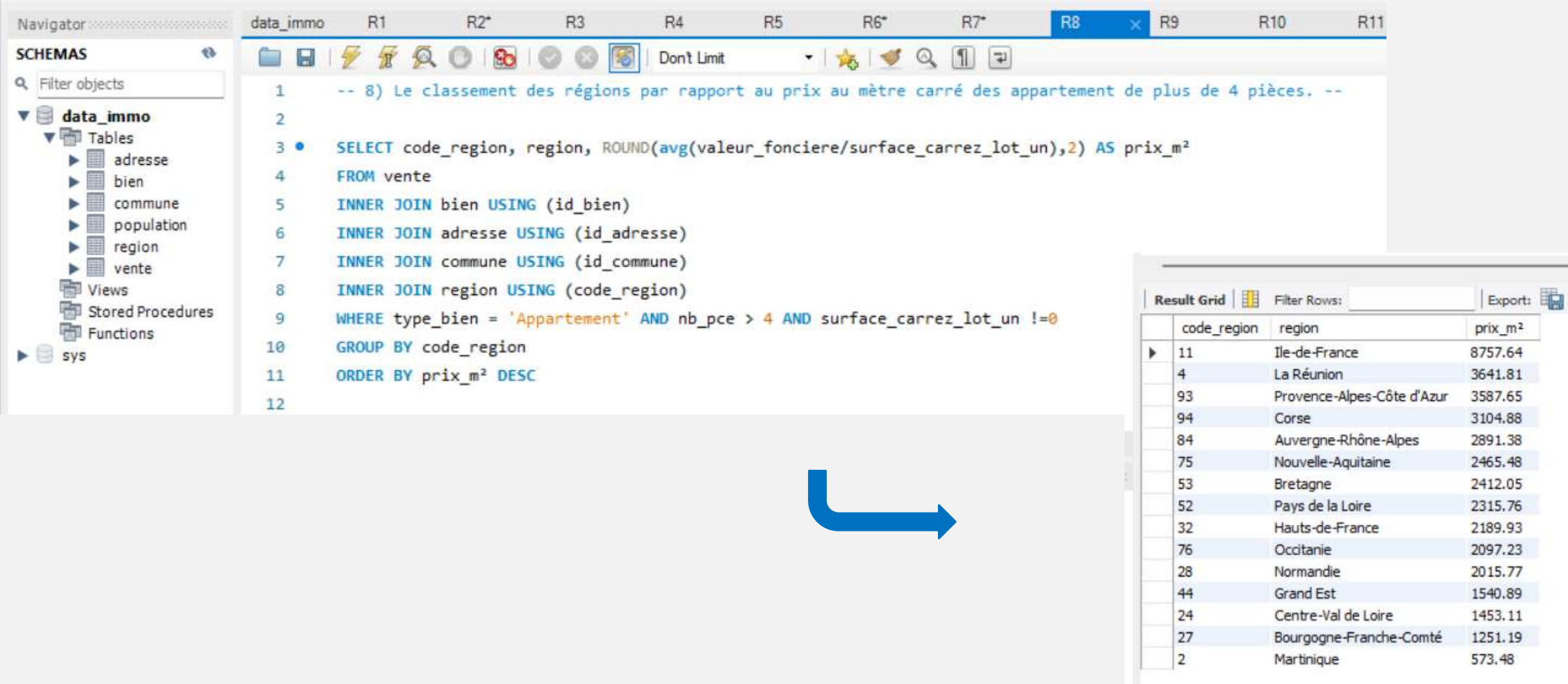
The screenshot displays a SQL IDE interface with a Navigator on the left and a main query editor. The Navigator shows a database schema named 'data\_immo' with tables: 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The main editor shows a SQL query with line numbers 1 through 17. The query calculates the percentage change in the number of sales between the first and second quarters of 2020. The result grid at the bottom shows a single value, 3.68.

```
1  -- 7) Taux d'évolution du nombre de ventes entre le premier et le second trimestre de 2020. --
2
3  WITH VENTE_1 AS (
4      SELECT
5          COUNT(id_vente) AS VENTE_TRIMESTRE_1
6      FROM vente
7      WHERE date_vente BETWEEN '2020-01-01' AND '2020-03-31'),
8
9  VENTE_2 AS (
10     SELECT
11         COUNT(id_vente) AS VENTE_TRIMESTRE_2
12     FROM vente
13     WHERE date_vente BETWEEN '2020-04-01' AND '2020-06-30')
14
15     SELECT ROUND((VENTE_TRIMESTRE_2-VENTE_TRIMESTRE_1)/VENTE_TRIMESTRE_1 * 100,2) AS Taux_évoluation_Nb_ventes
16     FROM VENTE_1, VENTE_2;
17
```

Result Grid

Taux_évoluation_Nb_ventes
3.68

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



The screenshot displays a database management system interface. On the left, a 'SCHEMAS' pane shows the 'data\_immo' database with tables like 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The main editor shows a SQL query (R8) that calculates the average price per square meter for apartments with more than 4 rooms, grouped by region. The query is as follows:

```
-- 8) Le classement des régions par rapport au prix au mètre carré des appartement de plus de 4 pièces. --  
  
SELECT code_region, region, ROUND(avg(valeur_fonciere/surface_carrez_lot_un),2) AS prix_m²  
FROM vente  
INNER JOIN bien USING (id_bien)  
INNER JOIN adresse USING (id_adresse)  
INNER JOIN commune USING (id_commune)  
INNER JOIN region USING (code_region)  
WHERE type_bien = 'Appartement' AND nb_pce > 4 AND surface_carrez_lot_un !=0  
GROUP BY code_region  
ORDER BY prix_m² DESC
```

The results are shown in a 'Result Grid' on the right, sorted by 'prix\_m²' in descending order:

	code_region	region	prix_m²
▶	11	Ile-de-France	8757.64
	4	La Réunion	3641.81
	93	Provence-Alpes-Côte d'Azur	3587.65
	94	Corse	3104.88
	84	Auvergne-Rhône-Alpes	2891.38
	75	Nouvelle-Aquitaine	2465.48
	53	Bretagne	2412.05
	52	Pays de la Loire	2315.76
	32	Hauts-de-France	2189.93
	76	Occitanie	2097.23
	28	Normandie	2015.77
	44	Grand Est	1540.89
	24	Centre-Val de Loire	1453.11
	27	Bourgogne-Franche-Comté	1251.19
	2	Martinique	573.48

## g) Liste des communes ayant eu au moins 50 ventes au 1er trimestre.

Navigator: data\_immo R1 R2\* R3 R4 R5 R6\* R7\* R8 R9\* x R10 R11 R12

SCHEMAS

Filter objects

data\_immo

- Tables
  - adresse
  - bien
  - commune
  - population
  - region
  - vente
- Views
- Stored Procedures
- Functions

sys

```
1 -- 9) Liste des communes ayant eu au moins 50 ventes au 1er trimestre. --
2
3 • SELECT no_dept, commune, COUNT(id_vente) AS Nombre_vente
4 FROM vente
5 INNER JOIN bien USING (id_bien)
6 INNER JOIN adresse USING (id_adresse)
7 INNER JOIN commune USING (id_commune)
8 WHERE date_vente BETWEEN '2020-01-01' AND '2020-03-31'
9 GROUP BY no_dept, commune
10 HAVING Nombre_vente > 50
11 ORDER BY Nombre_vente
12
```

Don't Limit

	no_dept	commune	Nombre_vente
▶	92	PUTEAUX	53
	2A	AJACCIO	54
	78	VERSAILLES	54
	94	SAINT-MAUR-DES-FOSSES	56
	92	LEVALLOIS-PERRET	59
	83	TOULON	59
	75	PARIS 4E ARRONDISSEMENT	60
	35	RENNES	61
	75	PARIS 2E ARRONDISSEMENT	61
	34	SETE	62
	13	LA CIOTAT	62
	75	PARIS 8E ARRONDISSEMENT	62
	30	NIMES	63
	49	ANGERS	64
	93	MONTREUIL	65
	13	MARSEILLE 9E ARRONDISSEMENT	66
	59	LILLE	67
	94	VINCENNES	68
	92	RUEIL-MALMAISON	68
	13	MARSEILLE 1ER ARRONDISSEMENT	71
	13	MARSEILLE 4E ARRONDISSEMENT	72
	6	ANTIBES	77

31	TOULOUSE	78
75	PARIS 3E ARRONDISSEMENT	79
92	ASNIERES-SUR-SEINE	79
75	PARIS 5E ARRONDISSEMENT	79
92	COURBEVOIE	80
13	MARSEILLE 8E ARRONDISSEMENT	81
75	PARIS 6E ARRONDISSEMENT	86
75	PARIS 7E ARRONDISSEMENT	87
75	PARIS 13E ARRONDISSEMENT	94
92	BOULOGNE-BILLANCOURT	99
75	PARIS 9E ARRONDISSEMENT	106
38	GRENOBLE	106
75	PARIS 10E ARRONDISSEMENT	109
75	PARIS 12E ARRONDISSEMENT	110
75	PARIS 19E ARRONDISSEMENT	116
44	NANTES	119
75	PARIS 20E ARRONDISSEMENT	127
75	PARIS 14E ARRONDISSEMENT	146
33	BORDEAUX	157
75	PARIS 16E ARRONDISSEMENT	165
75	PARIS 11E ARRONDISSEMENT	169
6	NICE	173
75	PARIS 18E ARRONDISSEMENT	209
75	PARIS 15E ARRONDISSEMENT	215
75	PARIS 17E ARRONDISSEMENT	228



## 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 IDE interface with a schema explorer on the left and a query editor on the right. The schema explorer shows a database named 'data\_immo' with tables: 'adresse', 'bien', 'commune', 'population', 'region', and 'vente'. The query editor contains the following SQL code:

```
-- 10) Différence en pourcentage du prix au mètre carré entre un appartement de 2 pièces et un appartement de 3 pièces. --  
  
WITH T2 AS (  
    SELECT ROUND(avg(valeur_fonciere/surface_carrez_lot_un),2) AS prix_m²_moyen_T2  
    FROM vente  
    INNER JOIN bien USING (id_bien)  
    WHERE nb_pce = 2 AND surface_carrez_lot_un != 0),  
  
T3 AS (  
    SELECT ROUND(avg(valeur_fonciere/surface_carrez_lot_un),2) AS prix_m²_moyen_T3  
    FROM vente  
    INNER JOIN bien USING (id_bien)  
    WHERE nb_pce = 3 AND surface_carrez_lot_un != 0)  
  
SELECT ROUND((prix_m²_moyen_T3-prix_m²_moyen_T2)/prix_m²_moyen_T2 * 100,2) AS Difference_T2_T3  
FROM T2, T3
```

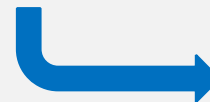
The result grid at the bottom shows the following data:

Difference_T2_T3
-12.8

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

```
Navigator
SCHEMAS
Filter objects
data_immo
Tables
  adresse
  bien
  commune
  population
  region
  vente
Views
Stored Procedures
Functions
sys

data_immo R1 R2* R3 R4 R5 R6* R7* R8 R9* R10 R11* R12*
1 -- 11) Les moyennes de valeurs foncières pour le top 3 des communes des départements 6, 13, 33, 59 et 69. --
2
3 WITH R11 AS (
4   SELECT CAST(no_dept AS decimal(3,0)) AS no_dept, departement, commune, ROUND(avg(valeur_fonciere),2) AS Moy_Valeur_fonciere
5   FROM vente
6   INNER JOIN bien USING (id_bien)
7   INNER JOIN adresse USING (id_adresse)
8   INNER JOIN commune USING (id_commune)
9   WHERE no_dept IN ('6','13','33','59','69') AND valeur_fonciere > 0
10  GROUP BY commune, no_dept, departement
11  ORDER BY Moy_Valeur_fonciere DESC),
12
13 R11B AS (
14   SELECT *, ROW_NUMBER() OVER (PARTITION BY no_dept ORDER BY Moy_Valeur_fonciere DESC) AS RAW
15   FROM R11)
16
17 SELECT no_dept, departement, commune, Moy_Valeur_fonciere
18 FROM R11B
19 WHERE RAW <= 3
20
```



Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	no_dept	departement	commune	Moy_Valeur_fonciere		
▶	6	Alpes-Maritimes	SAINT-JEAN-CAP-FERRAT	968750		
	6	Alpes-Maritimes	EZE	655000		
	6	Alpes-Maritimes	MOUANS-SARTOUX	476898.09		
	13	Bouches-du-Rhône	GIGNAC-LA-NERTHE	330000		
	13	Bouches-du-Rhône	SAINT-SAVOURNIN	314425		
	13	Bouches-du-Rhône	CASSIS	313416.88		
	33	Gironde	LEGE-CAP-FERRET	549500.64		
	33	Gironde	VAYRES	335000		
	33	Gironde	ARCACHON	307435.93		
	59	Nord	BERSEE	433202		
	59	Nord	CYSOING	408550		
	59	Nord	HALLUIN	322250		
	69	Rhône	VILLE-SUR-JARNIOUX	485300		
	69	Rhône	LYON 2E ARRONDISSEMENT	455217.27		
	69	Rhône	LYON 6E ARRONDISSEMENT	426968.25		

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

```
Navigator
SCHEMAS
Filter objects
data_immo
Tables
  adresse
  bien
  commune
  population
  region
  vente
Views
Stored Procedures
Functions
sys

data_immo R1 R2* R3 R4 R5 R6* R7* R8 R9* R10 R11* R12* x
1 -- 12) Les 20 communes avec le plus de transactions pour 1000 habitants pour les communes qui dépassent les 10 000 habitants. --
2
3 WITH R12 AS (
4   SELECT
5     no_dept,
6     departement,
7     commune,
8     COUNT(id_vente) AS Nombre_vente,
9     pop_tot
10  FROM vente
11  INNER JOIN bien USING (id_bien)
12  INNER JOIN adresse USING (id_adresse)
13  INNER JOIN commune USING (id_commune)
14  INNER JOIN population USING (id_pop)
15  WHERE pop_tot > 10000
16  GROUP BY no_dept, departement, commune, pop_tot)
17
18 SELECT *, ROUND((Nombre_vente*1000)/pop_tot,2) AS Ratio_pour_1000
19 FROM R12
20 ORDER BY Ratio_pour_1000 DESC
21 LIMIT 20
```

Result Grid						Filter Rows:	Export:	Wrap Cell Content:
	no_dept	departement	commune	Nombre_vente	pop_tot	Ratio_pour_1000		
▶	75	Paris	PARIS 2E ARRONDISSEMENT	127	21735	5.84		
	75	Paris	PARIS 1ER ARRONDISSEMENT	78	16055	4.86		
	75	Paris	PARIS 3E ARRONDISSEMENT	161	34306	4.69		
	33	Gironde	ARCACHON	55	11898	4.62		
	44	Loire-Atlantique	LA BAULE-ESCOUBLAC	77	16797	4.58		
	75	Paris	PARIS 4E ARRONDISSEMENT	120	29390	4.08		
	6	Alpes-Maritimes	ROQUEBRUNE-CAP-MARTIN	52	13041	3.99		
	75	Paris	PARIS 8E ARRONDISSEMENT	139	36250	3.83		
	83	Var	SANARY-SUR-MER	60	17160	3.50		
	83	Var	LA LONDE-LES-MAURES	37	10776	3.43		
	75	Paris	PARIS 9E ARRONDISSEMENT	208	60563	3.43		
	75	Paris	PARIS 6E ARRONDISSEMENT	139	41171	3.38		
	83	Var	SAINT-CYR-SUR-MER	38	11725	3.24		
	60	Oise	CHANTILLY	35	11178	3.13		
	44	Loire-Atlantique	PORNICHET	35	11440	3.06		
	94	Val-de-Marne	SAINT-MANDE	69	22576	3.06		
	75	Paris	PARIS 10E ARRONDISSEMENT	264	86863	3.04		
	6	Alpes-Maritimes	MENTON	91	30981	2.94		
	85	Vendée	SAINT-HILAIRE-DE-RIEZ	33	11501	2.87		
	94	Val-de-Marne	VINCENNES	141	50230	2.81		

