

P3 - Document avec les requêtes et les résultats  
SGBD utilisé : [MySQL Workbench](#)

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

```
SELECT
COUNT(*) AS Total_appart_1er_semestre
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
WHERE
Type_local = 'Appartement'
AND date_mutation BETWEEN '2020-01-01' AND '2020-06-30';
```

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, the SQL editor contains the following query:

```
1 • SELECT
2 COUNT(*) AS Total_appart_1er_semestre
3 FROM vente
4 JOIN bien
5 ON vente.id_bien = bien.id_bien
6 WHERE
7 Type_local = 'Appartement'
8 AND date_mutation BETWEEN '2020-01-01' AND '2020-06-30';
9
```

Below the SQL editor, the 'Result Grid' is visible. It shows a single column named 'Total\_appart\_1er\_semes...' with a value of 31378. The grid is labeled 'Result 1' and has a 'Read Only' status. To the right of the grid, there are icons for 'Result Grid' and 'Form Editor'.

At the bottom, the 'Action Output' panel shows the execution details:

	Time	Action	Response	Duration / Fetch Time
✓ 1	12:17:40	SELECT...	1 row(s) returned	0.084 sec / 0.00017 s...

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

```
SELECT
region.nom_reg,
count(*) AS Nombre_de_vente
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
JOIN region
ON departement.reg_code = region.code_reg
WHERE bien.Type_local = 'Appartement'
GROUP BY region.nom_reg
ORDER BY Nombre_de_vente DESC
```

The screenshot shows a database query tool interface. The top section, labeled 'Result Grid', displays the results of a SQL query. The grid has two columns: 'nom\_reg' and 'Nombre\_de\_ven...'. The data is sorted in descending order of 'Nombre\_de\_ven...'. The bottom section, labeled 'Action Output', shows a log of the query execution. It includes the SQL statement and the execution time for each step.

nom_reg	Nombre_de_ven...
Ile-de-France	13995
Provence-Alpes-Cote d'Azur	3649
Auvergne-Rhone-Alpes	3253
Nouvelle-Aquitaine	1932
Occitanie	1640
Pays de la Loire	1357
Hauts-de-France	1254
Grand Est	984
Bretagne	983
Normandie	862
Centre-Val de Loire	696
Bourgogne-Franche-Comte	376
Corse	223
Martinique	94
La Reunion	44
Guyane	34
Guadeloupe	2

Result 2

Action Output

	Time	Action	Response	Duration / Fetch Time
1	12:17:40	SELECT...	1 row(s) returned	0.084 sec / 0.00017 s...
2	12:21:37	SELECT r...	17 row(s) returned	0.450 sec / 0.000021...

```
1 • SELECT
2   region.nom_reg,
3   count(*) AS Nombre_de_vente
4 FROM vente
5 JOIN bien
6   ON vente.id_bien = bien.id_bien
7 JOIN commune
8   ON bien.Code_ID_commune = commune.Code_ID_commune
9 JOIN departement
10  ON departement.code_dep = commune.Code_dep
11 JOIN region
12  ON departement.reg_code = region.code_reg
13 WHERE bien.Type_local = 'Appartement'
14 GROUP BY region.nom_reg
15 ORDER BY Nombre_de_vente DESC
16
17
```

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

```
WITH
Nombre_Total_Appartement AS (
SELECT
COUNT(*) AS Total_appart_1er_semestre
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
WHERE
Type_local = 'Appartement'
AND date_mutation BETWEEN '2020-01-01' AND '2020-06-30')

SELECT
bien.nb_pieces,
round(count(*)/(Select * from Nombre_Total_Appartement)*100,2) AS Proportion
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
WHERE bien.Type_local = 'Appartement'
GROUP BY bien.nb_pieces
ORDER BY bien.nb_pieces
```

Limit to 1000 rows

```
1 • WITH
2   Nombre_Total_Appartement AS (
3     SELECT
4       COUNT(*) AS Total_appart_1er_semestre
5     FROM vente
6     JOIN bien
7     ON vente.id_bien = bien.id_bien
8     WHERE
9       Type_local = 'Appartement'
10    AND date_mutation BETWEEN '2020-01-01' AND '2020-06-30')
11
12    SELECT
13      bien.nb_pieces,
14      round(count(*)/(Select * from Nombre_Total_Appartement)*100,2) AS Proport:
15    FROM vente
16    JOIN bien
17    ON vente.id_bien = bien.id_bien
18    WHERE bien.Type_local = 'Appartement'
19    GROUP BY bien.nb_pieces
20    ORDER BY bien.nb_pieces
21
```

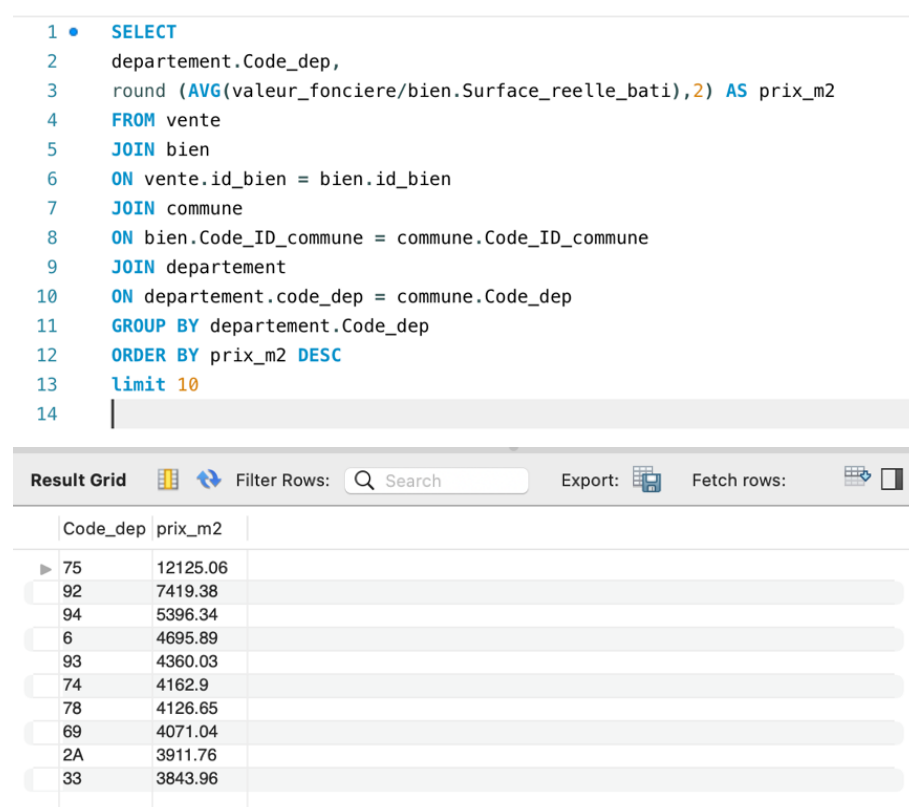
nb_pieces	Proportion
0	0.10
1	21.48
2	31.18
3	28.57
4	14.21
5	3.55
6	0.65
7	0.17
8	0.05
9	0.03
10	0.01
11	0.00

Result 4

Read Only

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

```
SELECT
departement.Code_dep,
round (AVG(valeur_fonciere/bien.Surface_reelle_bati),2) AS prix_m2
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
GROUP BY departement.Code_dep
ORDER BY prix_m2 DESC
limit 10
```



```
1 • SELECT
2   departement.Code_dep,
3   round (AVG(valeur_fonciere/bien.Surface_reelle_bati),2) AS prix_m2
4 FROM vente
5 JOIN bien
6 ON vente.id_bien = bien.id_bien
7 JOIN commune
8 ON bien.Code_ID_commune = commune.Code_ID_commune
9 JOIN departement
10 ON departement.code_dep = commune.Code_dep
11 GROUP BY departement.Code_dep
12 ORDER BY prix_m2 DESC
13 limit 10
14
```

Code_dep	prix_m2
75	12125.06
92	7419.38
94	5396.34
6	4695.89
93	4360.03
74	4162.9
78	4126.65
69	4071.04
2A	3911.76
33	3843.96

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

```
SELECT
region.nom_reg,
round (avg(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS Prix_moyen_m2
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
JOIN region
ON departement.reg_code = region.code_reg
WHERE bien.Type_local = 'Maison' AND nom_reg = 'Ile-de-France'
```

The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and navigation. The query text is as follows:

```
4 FROM vente
5 JOIN bien
6 ON vente.id_bien = bien.id_bien
7 JOIN commune
8 ON bien.Code_ID_commune = commune.Code_ID_commune
9 JOIN departement
10 ON departement.code_dep = commune.Code_dep
11 JOIN region
12 ON departement.reg_code = region.code_reg
13 WHERE bien.Type_local = 'Maison' AND nom_reg = 'Ile-de-France'
14
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' search bar and an 'Export' button. The grid displays the following data:

nom_reg	Prix_moyen_m2
Ile-de-France	3993.15

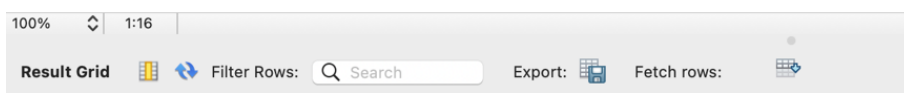
At the bottom of the interface, it indicates 'Result 6' and a 'Read Only' status.

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

```
SELECT
bien.id_bien, bien.Surface_reelle_bati, vente.valeur_fonciere
AS Prix_appartements, region.nom_reg
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
JOIN region
ON departement.reg_code = region.code_reg
WHERE Type_local = 'Appartement'
ORDER BY Prix_appartements DESC
LIMIT 10
```



```
1 • SELECT
2 bien.id_bien, bien.Surface_reelle_bati, vente.valeur_fonciere
3 AS Prix_appartements, region.nom_reg
4 FROM vente
5 JOIN bien
6 ON vente.id_bien = bien.id_bien
7 JOIN commune
8 ON bien.Code_ID_commune = commune.Code_ID_commune
9 JOIN departement
10 ON departement.code_dep = commune.Code_dep
11 JOIN region
12 ON departement.reg_code = region.code_reg
13 WHERE Type_local = 'Appartement'
14 ORDER BY Prix_appartements DESC
15 LIMIT 10
16 |
```



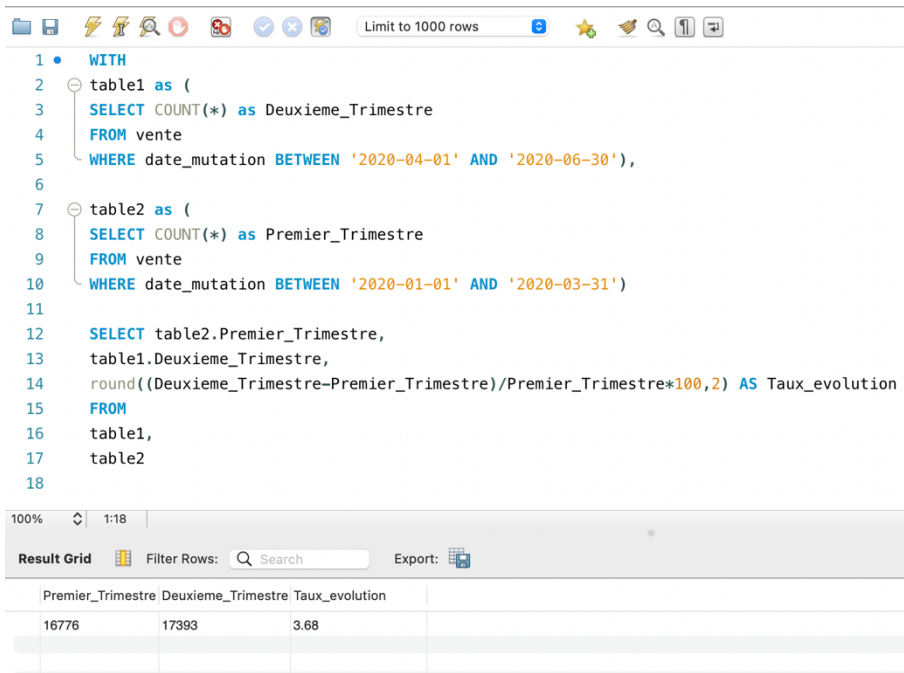
	id_bien	Surface_reelle_b...	Prix_apparteme...	nom_reg
▶	30570	10	9000000	Ile-de-France
	20662	62	8600000	Ile-de-France
	28194	289	8577713	Ile-de-France
	30723	42	7620000	Ile-de-France
	28245	200	7600000	Ile-de-France
	27927	143	7535000	Ile-de-France
	30278	357	7420000	Ile-de-France
	30436	241	7200000	Ile-de-France
	27762	310	7050000	Ile-de-France
	27919	76	6600000	Ile-de-France

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

```
WITH
table1 as (
SELECT COUNT(*) as Deuxieme_Trimestre
FROM vente
WHERE date_mutation BETWEEN '2020-04-01' AND '2020-06-30'),

table2 as (
SELECT COUNT(*) as Premier_Trimestre
FROM vente
WHERE date_mutation BETWEEN '2020-01-01' AND '2020-03-31')

SELECT table2.Premier_Trimestre,
table1.Deuxieme_Trimestre,
round((Deuxieme_Trimestre-Premier_Trimestre)/Premier_Trimestre*100,2) AS Taux_evolution
FROM
table1,
table2
```

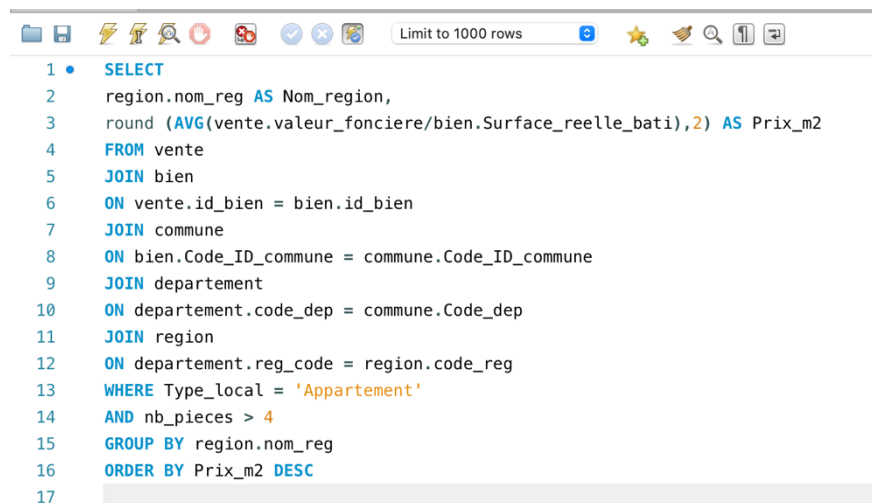


```
1 • WITH
2   table1 as (
3     SELECT COUNT(*) as Deuxieme_Trimestre
4     FROM vente
5     WHERE date_mutation BETWEEN '2020-04-01' AND '2020-06-30'),
6
7   table2 as (
8     SELECT COUNT(*) as Premier_Trimestre
9     FROM vente
10    WHERE date_mutation BETWEEN '2020-01-01' AND '2020-03-31')
11
12  SELECT table2.Premier_Trimestre,
13         table1.Deuxieme_Trimestre,
14         round((Deuxieme_Trimestre-Premier_Trimestre)/Premier_Trimestre*100,2) AS Taux_evolution
15  FROM
16  table1,
17  table2
18
```

Premier_Trimestre	Deuxieme_Trimestre	Taux_evolution
16776	17393	3.68

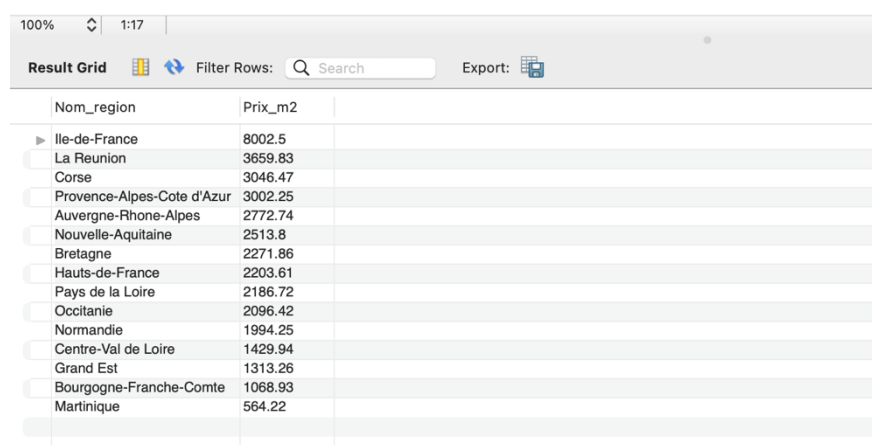
## 8. Le classement des régions par rapport au prix au mètre carré des appartement de plus de 4 pièces.

```
SELECT
region.nom_reg AS Nom_region,
round (AVG(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS Prix_m2
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
JOIN region
ON departement.reg_code = region.code_reg
WHERE Type_local = 'Appartement'
AND nb_pieces > 4
GROUP BY region.nom_reg
ORDER BY Prix_m2 DESC
```



The screenshot shows a SQL editor with a toolbar at the top containing icons for file operations, search, and execution. Below the toolbar, the SQL query is displayed in a monospaced font. The query is a SELECT statement that calculates the average price per square meter for apartments with more than 4 rooms, grouped by region. The results are ordered in descending order of price per square meter. The query is as follows:

```
1 • SELECT
2   region.nom_reg AS Nom_region,
3   round (AVG(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS Prix_m2
4 FROM vente
5 JOIN bien
6 ON vente.id_bien = bien.id_bien
7 JOIN commune
8 ON bien.Code_ID_commune = commune.Code_ID_commune
9 JOIN departement
10 ON departement.code_dep = commune.Code_dep
11 JOIN region
12 ON departement.reg_code = region.code_reg
13 WHERE Type_local = 'Appartement'
14 AND nb_pieces > 4
15 GROUP BY region.nom_reg
16 ORDER BY Prix_m2 DESC
17
```



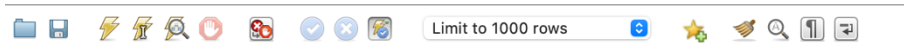
The screenshot shows the result grid of the SQL query. The interface includes a toolbar with a 'Filter Rows' button and a search bar. The result grid displays the following data:

Nom_region	Prix_m2
Ile-de-France	8002.5
La Reunion	3659.83
Corse	3046.47
Provence-Alpes-Cote d'Azur	3002.25
Auvergne-Rhone-Alpes	2772.74
Nouvelle-Aquitaine	2513.8
Bretagne	2271.86
Hauts-de-France	2203.61
Pays de la Loire	2186.72
Occitanie	2096.42
Normandie	1994.25
Centre-Val de Loire	1429.94
Grand Est	1313.26
Bourgogne-Franche-Comte	1068.93
Martinique	564.22

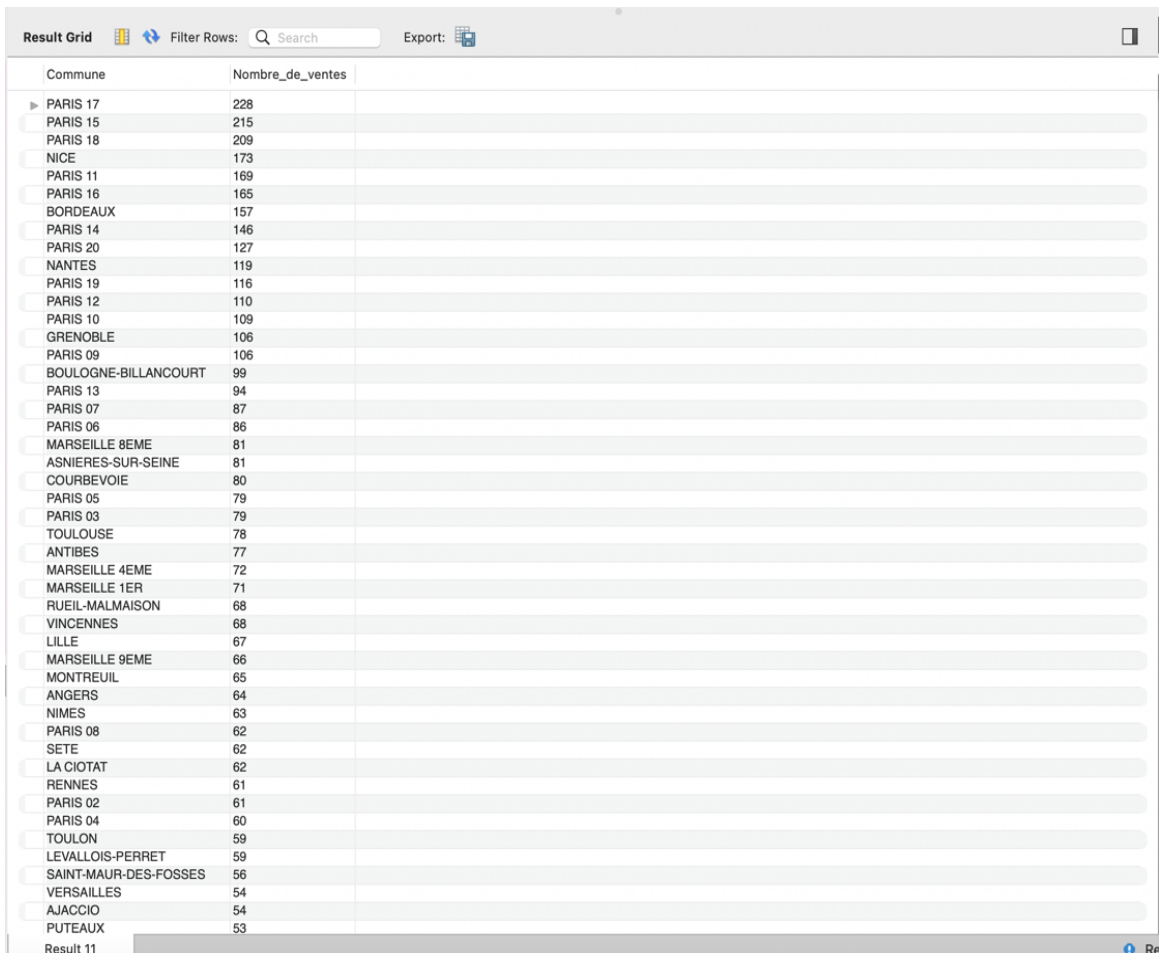


## 9. Liste des communes ayant eu au moins 50 ventes au 1er trimestre

```
SELECT
commune.Commune,
count(id_vente) AS Nombre_de_ventes
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
WHERE vente.date_mutation BETWEEN '2020-01-01' AND '2020-03-31'
GROUP BY commune.Commune
HAVING Nombre_de_ventes >= 50
ORDER BY Nombre_de_ventes DESC
```



```
1 • SELECT
2     commune.Commune,
3     count(id_vente) AS Nombre_de_ventes
4 FROM vente
5 JOIN bien
6 ON vente.id_bien = bien.id_bien
7 JOIN commune
8 ON bien.Code_ID_commune = commune.Code_ID_commune
9 WHERE vente.date_mutation BETWEEN '2020-01-01' AND '2020-03-31'
10 GROUP BY commune.Commune
11 HAVING Nombre_de_ventes >= 50
12 ORDER BY Nombre_de_ventes DESC
13
```



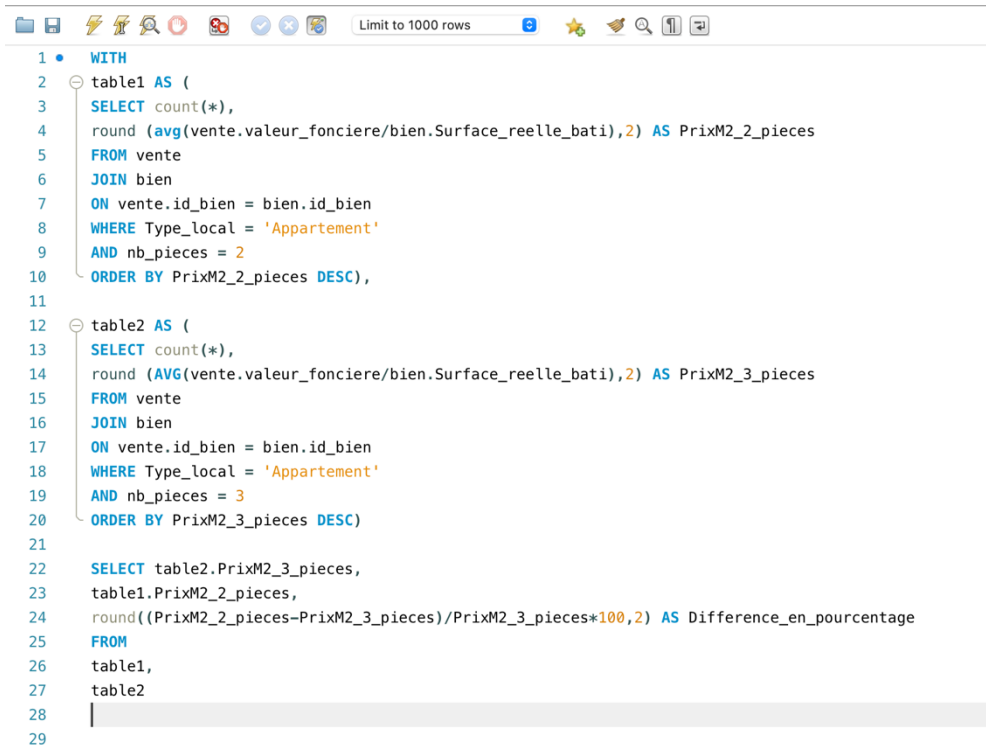
Commune	Nombre_de_ventes
PARIS 17	228
PARIS 15	215
PARIS 18	209
NICE	173
PARIS 11	169
PARIS 16	165
BORDEAUX	157
PARIS 14	146
PARIS 20	127
NANTES	119
PARIS 19	116
PARIS 12	110
PARIS 10	109
GRENOBLE	106
PARIS 09	106
BOULOGNE-BILLANCOURT	99
PARIS 13	94
PARIS 07	87
PARIS 06	86
MARSEILLE 8EME	81
ASNIERES-SUR-SEINE	81
COURBEVOIE	80
PARIS 05	79
PARIS 03	79
TOULOUSE	78
ANTIBES	77
MARSEILLE 4EME	72
MARSEILLE 1ER	71
RUEIL-MALMAISON	68
VINCENNES	68
LILLE	67
MARSEILLE 9EME	66
MONTREUIL	65
ANGERS	64
NIMES	63
PARIS 08	62
SETE	62
LA CIOTAT	62
RENNES	61
PARIS 02	61
PARIS 04	60
TOULON	59
LEVALLOIS-PERRET	59
SAINT-MAUR-DES-FOSSES	56
VERSAILLES	54
AJACCIO	54
PUTEAUX	53

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

```
WITH
table1 AS (
SELECT count(*),
round (avg(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS PrixM2_2_pieces
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
WHERE Type_local = 'Appartement'
AND nb_pieces = 2
ORDER BY PrixM2_2_pieces DESC),

table2 AS (
SELECT count(*),
round (AVG(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS PrixM2_3_pieces
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
WHERE Type_local = 'Appartement'
AND nb_pieces = 3
ORDER BY PrixM2_3_pieces DESC)

SELECT table2.PrixM2_3_pieces,
table1.PrixM2_2_pieces,
round((PrixM2_2_pieces-PrixM2_3_pieces)/PrixM2_3_pieces*100,2) AS Difference_en_pourcentage
FROM
table1,
table2
```



```
1 • WITH
2 table1 AS (
3 SELECT count(*),
4 round (avg(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS PrixM2_2_pieces
5 FROM vente
6 JOIN bien
7 ON vente.id_bien = bien.id_bien
8 WHERE Type_local = 'Appartement'
9 AND nb_pieces = 2
10 ORDER BY PrixM2_2_pieces DESC),
11
12 table2 AS (
13 SELECT count(*),
14 round (AVG(vente.valeur_fonciere/bien.Surface_reelle_bati),2) AS PrixM2_3_pieces
15 FROM vente
16 JOIN bien
17 ON vente.id_bien = bien.id_bien
18 WHERE Type_local = 'Appartement'
19 AND nb_pieces = 3
20 ORDER BY PrixM2_3_pieces DESC)
21
22 SELECT table2.PrixM2_3_pieces,
23 table1.PrixM2_2_pieces,
24 round((PrixM2_2_pieces-PrixM2_3_pieces)/PrixM2_3_pieces*100,2) AS Difference_en_pourcentage
25 FROM
26 table1,
27 table2
28
29
```

100% 1:28

Result Grid Filter Rows: Search Export:

	PrixM2_3_pieces	PrixM2_2_piec...	Difference_en_pourcent...
▶	4285.28	4930.41	15.05

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

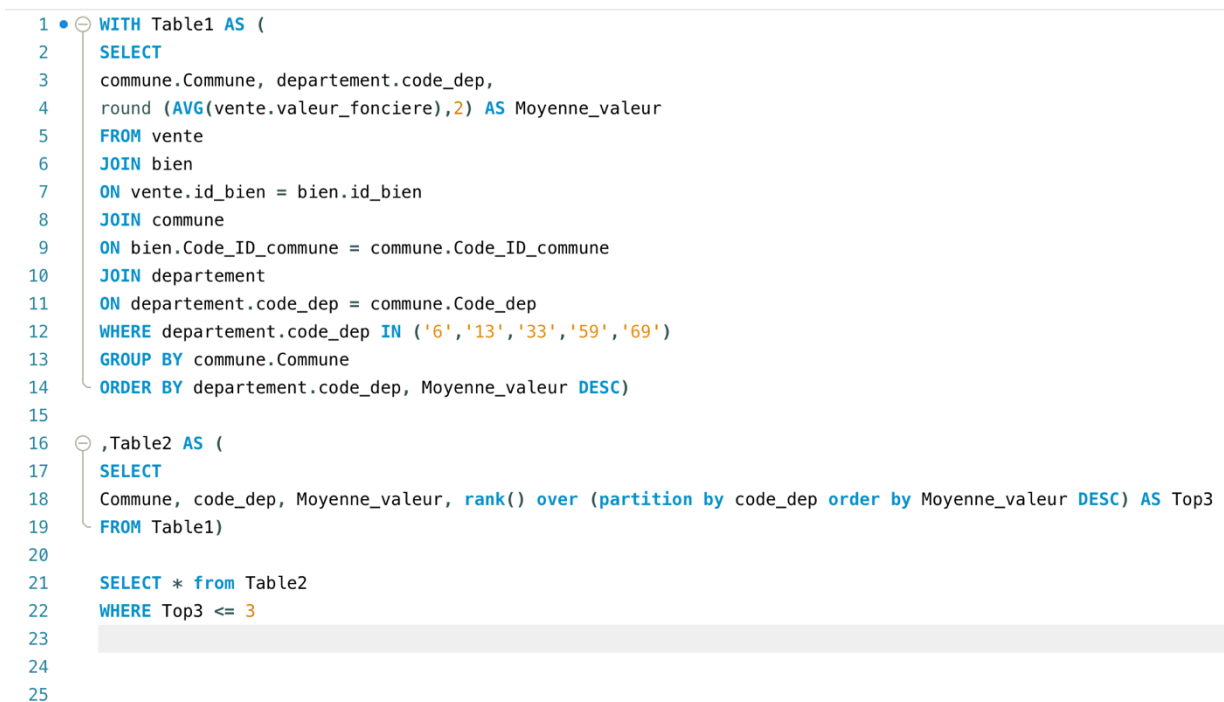
```

WITH Table1 AS (
SELECT
commune.Commune, departement.code_dep,
round (AVG(vente.valeur_fonciere),2) AS Moyenne_valeur
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
WHERE departement.code_dep IN ('6','13','33','59','69')
GROUP BY commune.Commune
ORDER BY departement.code_dep, Moyenne_valeur DESC)

,Table2 AS (
SELECT
Commune, code_dep, Moyenne_valeur, rank() over (partition by code_dep order by Moyenne_valeur DESC) AS Top3
FROM Table1)

SELECT * from Table2
WHERE Top3 <= 3

```



## 2 option pour la requête 11 (Moyenne unique pour chaque département)

```
WITH Table1 AS (
SELECT
commune.Commune, departement.code_dep,
round (AVG(vente.valeur_fonciere),2) AS Moyenne_valeur
FROM vente
JOIN bien
ON vente.id_bien = bien.id_bien
JOIN commune
ON bien.Code_ID_commune = commune.Code_ID_commune
JOIN departement
ON departement.code_dep = commune.Code_dep
```

```
,Table2 AS (
SELECT
Commune, code_dep, Moyenne_valeur, rank() over (partition by code_dep order by Moyenne_valeur DESC) AS Top3
FROM Table1)
```



















00% 1:26

[illegible]