

Requête 1 : Nombre total d'appartements vendus au 1er semestre 2020

The screenshot shows the MySQL Workbench interface. The central editor displays the following SQL query:

```
1 SELECT DISTINCT b.Type_local
2 FROM projet3.bien b;
3 SELECT COUNT(b.Type_local) AS nombre_total_appart
4 FROM projet3.bien b
5 JOIN projet3.vente v USING(id_bien)
6 WHERE lower(b.Type_local) = "appartement"
7 AND v.Date between "2020-01-01" AND "2020-06-30";
```

The query is executed, and the results are shown in the 'Result Grid' at the bottom. The grid has two columns: 'nombre_total_appart' and a value of 31378.

On the left, the 'SCHEMAS' pane shows the database structure. The 'Table: vente' is detailed with the following columns:

Columns:	
id_vente	int PK
id_bien	int
Date	date
Valeur	float

The bottom status bar shows the system temperature as 0°C and the date as 16/12/2022.

Requête 2 : Le nombre de ventes d'appartement par région pour le 1er semestre

MySQL Workbench

Navigation: project3 requête 2

SQL Editor:

```
1 SELECT
2   r.id_region,
3   r.nom_region,
4   count(r.id_region) AS Nombre_vente_appart
5 FROM projet3.vente v
6 JOIN projet3.bien b USING(id_bien)
7 JOIN projet3.commune c USING(id_codedep_codecommune)
8 JOIN projet3.departement d USING(id_departement)
9 JOIN projet3.region r USING(id_region)
10 WHERE lower(b.Type_local) = "Appartement"
11 AND v.Date between "2020-01-01" AND "2020-06-30"
12 GROUP BY r.id_region
13 ORDER BY Nombre_vente_appart DESC
```

Result Grid:

id_region	nom_region	Nombre_vente_appart
11	Ile-de-France	13995
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
02	Martinique	94
04	La Réunion	44
03	Guyane	34

Table: vente

Columns:

- id_vente: int PK
- id_bien: int
- Date: date
- Valeur: float

Object Info: Session

System: 0°C Temps dégaigé 16/12/2022

MySQL Workbench

Navigation: project3 requête 2

SQL Editor:

```
1 SELECT
2   r.id_region,
3   r.nom_region,
4   count(r.id_region) AS Nombre_vente_appart
5 FROM projet3.vente v
```

Result Grid:

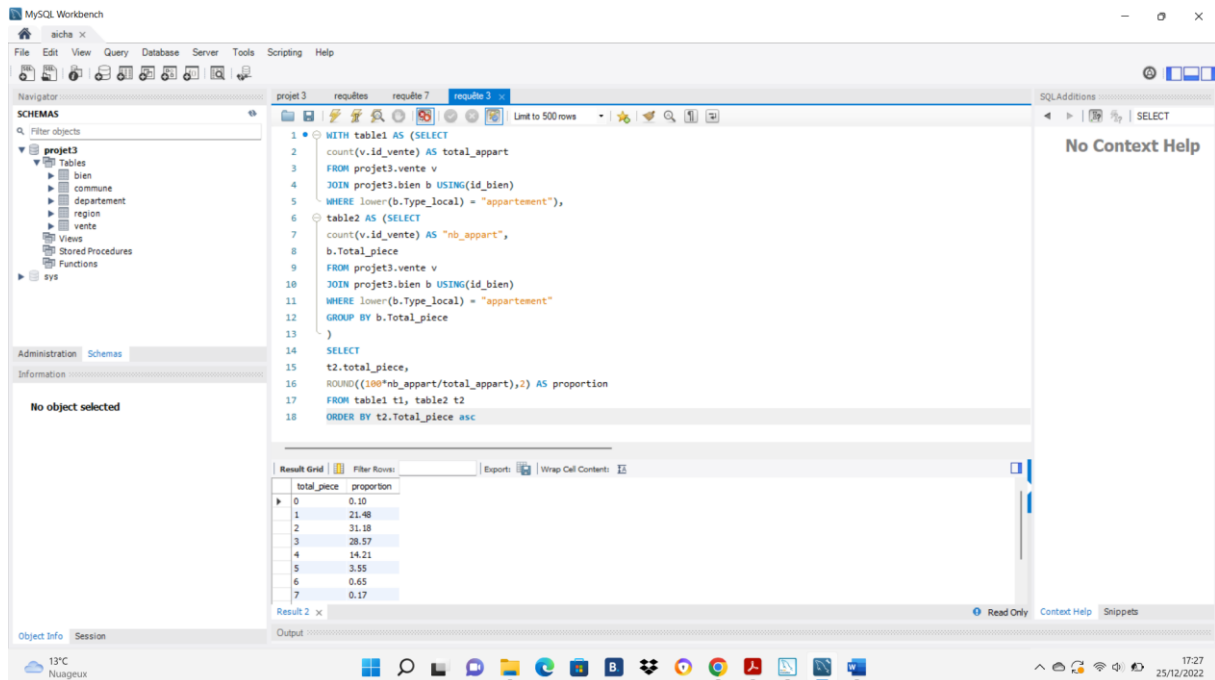
id_region	nom_region	Nombre_vente_appart
11	Ile-de-France	13995
93	Provence-Alpes-Côte d'Azur	3649
84	Auvergne-Rhône-Alpes	3253
75	Nouvelle-Aquitaine	1932
76	Occitanie	1640
52	Pays de la Loire	1357
32	Hauts-de-France	1254
44	Grand Est	984
53	Bretagne	983
28	Normandie	862
24	Centre-Val de Loire	696
27	Bourgogne-Franche-Comté	376
94	Corse	223
02	Martinique	94
04	La Réunion	44
03	Guyane	34
01	Guadeloupe	2

No object selected

Object Info: Session

System: -2°C Ciel couvert 16/12/2022

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



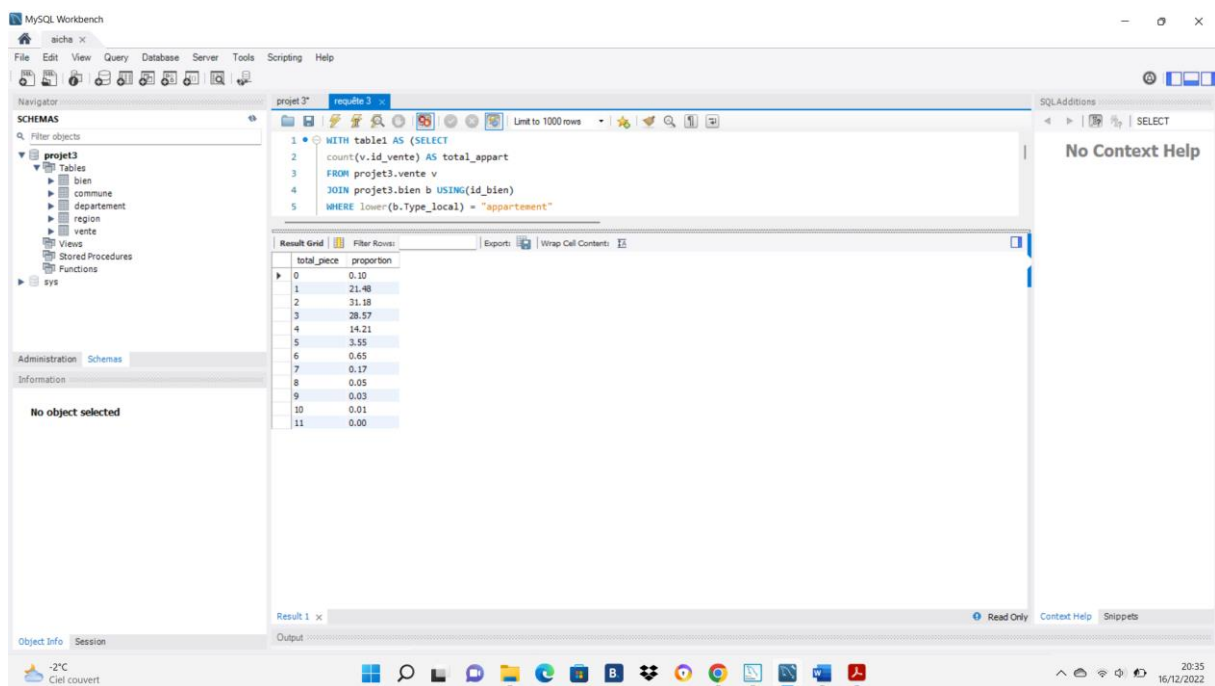
MySQL Workbench interface showing the execution of a SQL query (Requête 3) to calculate the proportion of sales of apartments by the number of rooms.

The query is as follows:

```
1 WITH table1 AS (SELECT
2   count(v.id_vente) AS total_appart
3 FROM projet3.vente v
4 JOIN projet3.bien b USING(id_bien)
5 WHERE lower(b.Type_local) = "appartement"),
6 table2 AS (SELECT
7   count(v.id_vente) AS "nb_appart",
8   b.Total_piece
9 FROM projet3.vente v
10 JOIN projet3.bien b USING(id_bien)
11 WHERE lower(b.Type_local) = "appartement"
12 GROUP BY b.Total_piece
13 )
14 SELECT
15   t2.total_piece,
16   ROUND((100*nb_appart/total_appart),2) AS proportion
17 FROM table1 t1, table2 t2
18 ORDER BY t2.Total_piece asc
```

The result grid displays the following data:

total_piece	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



MySQL Workbench interface showing the execution of a simplified SQL query (Requête 3) to calculate the proportion of sales of apartments by the number of rooms.

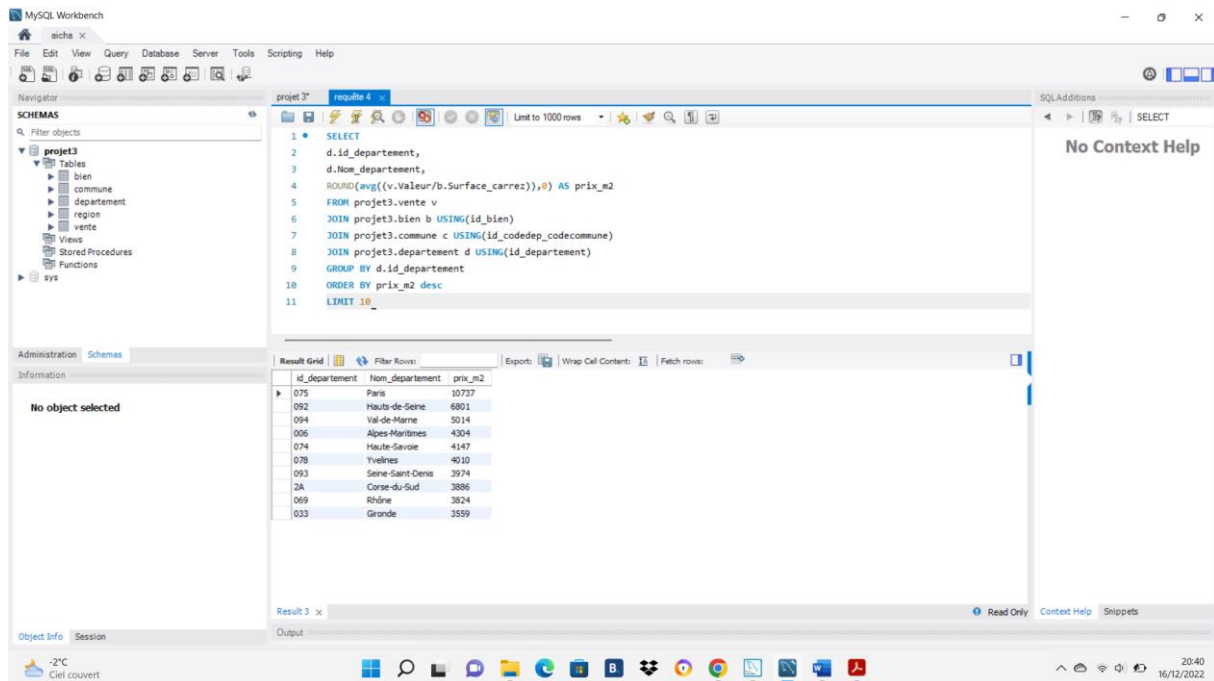
The query is as follows:

```
1 WITH table1 AS (SELECT
2   count(v.id_vente) AS total_appart
3 FROM projet3.vente v
4 JOIN projet3.bien b USING(id_bien)
5 WHERE lower(b.Type_local) = "appartement")
```

The result grid displays the following data:

total_piece	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

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



The screenshot shows the MySQL Workbench interface. The central pane displays a SQL query for 'requête 4' which selects department information and calculates the average price per square meter. The results are shown in a table below the query.

```
1 SELECT
2   d.id_departement,
3   d.Nom_departement,
4   ROUND(avg((v.Valeur/b.Surface_carrez)),0) AS prix_m2
5 FROM projet3.vente v
6 JOIN projet3.bien b USING(id_bien)
7 JOIN projet3.commune c USING(id_codedep_codecommune)
8 JOIN projet3.departement d USING(id_departement)
9 GROUP BY d.id_departement
10 ORDER BY prix_m2 desc
11 LIMIT 10
```

id_departement	Nom_departement	prix_m2
075	Paris	10737
092	Hauts-de-Seine	6801
094	Val-de-Marne	5014
006	Alpes-Maritimes	4304
074	Haute-Savoie	4147
078	Yvelines	4010
093	Seine-Saint-Denis	3974
2A	Corse-du-Sud	3886
069	Rhône	3824
033	Gironde	3559

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

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'bien', 'commune', 'departement', 'region', and 'vente'. The main editor window contains a SQL query (Query 5) that calculates the average price per square meter for houses in the Île-de-France region. The query is as follows:

```
1 SELECT
2   r.id_region
3 FROM projet3.region r
4 WHERE lower(nom_region) LIKE "ile%";
5 SELECT DISTINCT b.type_local
6 FROM projet3.bien b;
7 SELECT
8   r.id_region,
9   r.Nom_region,
10  ROUND(avg(v.Valeur/b.Surface_carrez),0) AS prix_moy_m2_maison
11 FROM projet3.vente v
12 JOIN projet3.bien b USING(id_bien)
13 JOIN projet3.commune c USING(id_codedep_codecommune)
14 JOIN projet3.departement d USING(id_departement)
15 JOIN projet3.region r USING(id_region)
16 WHERE lower(b.Type_local) = "maison"
17 AND r.id_region = 11
```

Below the query, the 'Result Grid' shows the output of the query:

id_region	Nom_region	prix_moy_m2_maison
11	Île-de-France	3702

The bottom status bar indicates the current session is 'region 4 bien 5 Result 6 x' and the output is displayed. The system tray at the bottom shows the date and time as 20:42 on 16/12/2022.

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

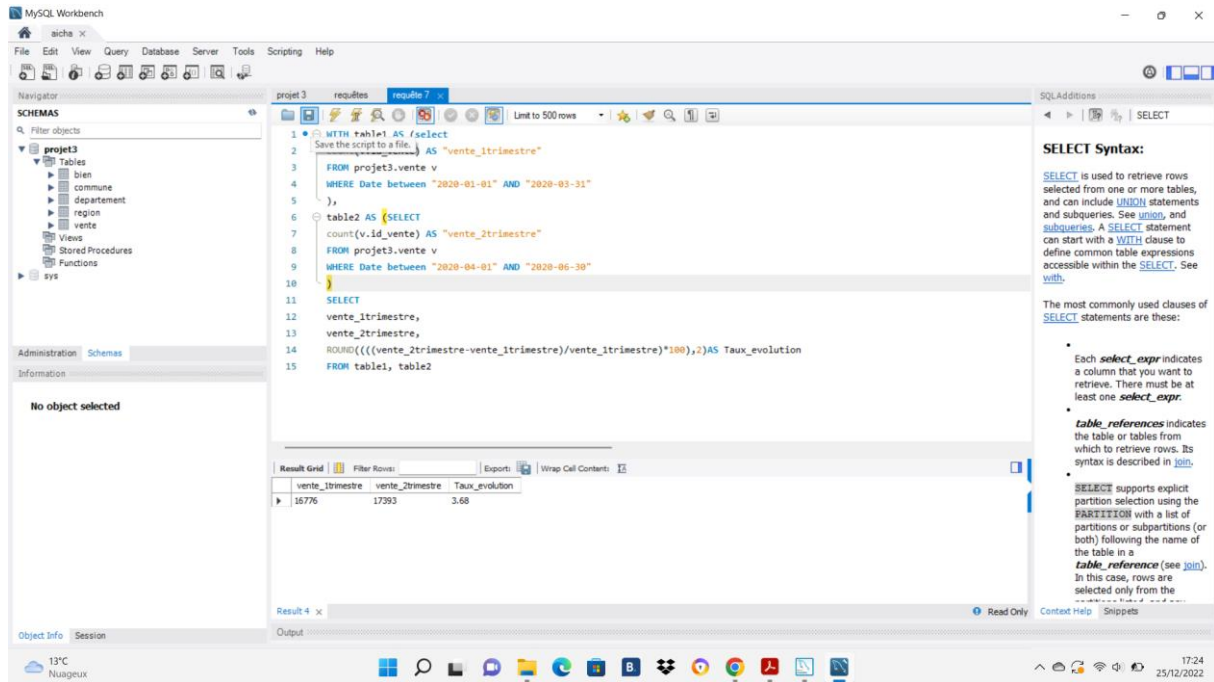
The screenshot shows the MySQL Workbench interface. The central editor displays a SQL query (Requête 6) that selects the top 10 most expensive apartments based on their value, including details about the property, department, region, and surface area.

```
1 SELECT
2   b.id_bien,
3   d.id_departement,
4   d.Nom_departement,
5   r.id_region,
6   r.Nom_region,
7   b.Surface_carrez,
8   v.Valeur
9 FROM projet3.vente v
10 JOIN projet3.bien b USING(id_bien)
11 JOIN projet3.commune c USING(id_codedep_codecommune)
12 JOIN projet3.departement d USING(id_departement)
13 JOIN projet3.region r USING(id_region)
14 WHERE lower(b.Type_local) = "appartement"
15 ORDER BY v.Valeur desc
16 LIMIT 10
```

Below the query editor, the 'Result Grid' shows the results of the query. The results are sorted by value in descending order, showing the top 10 most expensive apartments.

id_bien	id_departement	Nom_departement	id_region	Nom_region	Surface_carrez	Valeur
32275	075	Paris	11	Ile-de-France	44570	9000000
21835	091	Essonne	11	Ile-de-France	64	8600000
29799	075	Paris	11	Ile-de-France	20.55	8577710
32433	075	Paris	11	Ile-de-France	42.77	7620000
29850	075	Paris	11	Ile-de-France	253.3	7600000
29522	075	Paris	11	Ile-de-France	139.9	7535000
31973	075	Paris	11	Ile-de-France	360.95	7420000
32135	075	Paris	11	Ile-de-France	595	7200000
29353	075	Paris	11	Ile-de-France	122.56	7050000
29513	075	Paris	11	Ile-de-France	79.38	6600000

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



The screenshot shows the MySQL Workbench interface. The central pane displays a SQL query for 'Requête 7'. The query uses Common Table Expressions (CTEs) to calculate the percentage change in sales volume between the first and second quarters of 2020. The left sidebar shows the 'project3' database schema with tables like 'bien', 'commune', 'departement', 'region', 'vente', 'Views', 'Stored Procedures', and 'Functions'. The bottom pane shows the 'Result Grid' with one row of data.

```
1 WITH table1 AS (select
2   count(v.vente) AS "vente_1trimestre"
3   FROM projet3.vente v
4   WHERE Date between "2020-01-01" AND "2020-03-31"
5 ),
6 table2 AS (SELECT
7   count(v.id_vente) AS "vente_2trimestre"
8   FROM projet3.vente v
9   WHERE Date between "2020-04-01" AND "2020-06-30"
10 )
11 SELECT
12   vente_1trimestre,
13   vente_2trimestre,
14   ROUND((((vente_2trimestre-vente_1trimestre)/vente_1trimestre)*100),2) AS Taux_evolution
15 FROM table1, table2
```

vente_1trimestre	vente_2trimestre	Taux_evolution
16776	17393	3.68

Result 4 x

Output

13°C
Nuageux

17:24
25/12/2022

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

The screenshot shows the MySQL Workbench interface with a query window titled 'requête 8'. The query is as follows:

```
1 SELECT
2   r.id_region,
3   r.Nom_region,
4   b.Total_piece,
5   ROUND(avg(v.valeur/b.Surface_carrez),0) AS prix_m2
6 FROM projet3.vente v
7 JOIN projet3.bien b USING(id_bien)
8 JOIN projet3.commune c USING(id_codedep_codecommune)
9 JOIN projet3.departement d USING(id_departement)
10 JOIN projet3.region r USING(id_region)
11 WHERE lower(b.Type_local) = "appartement"
12 AND b.Total_piece > 4
13 GROUP BY r.id_region
14 ORDER BY prix_m2 desc
```

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

id_region	Nom_region	Total_piece	prix_m2
11	Ile-de-France	5	8751
04	La Réunion	5	3642
93	Provence-Alpes-Côte d'Azur	5	3588
94	Corse	5	3105
84	Auvergne-Rhône-Alpes	6	2891
75	Nouvelle-Aquitaine	5	2465
53	Bretagne	5	2412
52	Pays de la Loire	5	2299
32	Hauts-de-France	6	2190
76	Occitanie	5	2097
28	Normandie	5	2016
44	Grand Est	6	1541

The screenshot shows the MySQL Workbench interface with a query window titled 'requête 8'. The query is as follows:

```
1 SELECT
2   r.id_region,
3   r.Nom_region,
4   b.Total_piece,
5   ROUND(avg(v.valeur/b.Surface_carrez),0) AS prix_m2
6 FROM projet3.vente v
```

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

id_region	Nom_region	Total_piece	prix_m2
11	Ile-de-France	5	8751
04	La Réunion	5	3642
93	Provence-Alpes-Côte d'Azur	5	3588
94	Corse	5	3105
84	Auvergne-Rhône-Alpes	6	2891
75	Nouvelle-Aquitaine	5	2465
53	Bretagne	5	2412
52	Pays de la Loire	5	2299
32	Hauts-de-France	6	2190
76	Occitanie	5	2097
28	Normandie	5	2016
44	Grand Est	6	1541
24	Centre-val de Loire	5	1453
27	Bourgogne-Franche-Comté	5	1251
02	Martinique	5	573

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'project3' selected. The central editor contains the following SQL query:

```
1 SELECT
2   b.id_codedep_codecommune,
3   c.Nom_commune,
4   count(id_vente) AS "ventes"
5 FROM projet3.vente v
6 JOIN projet3.bien b USING(id_bien)
7 JOIN projet3.commune c USING(id_codedep_codecommune)
8 WHERE v.Date between "2020-01-01" AND "2020-03-31"
9 GROUP BY b.id_codedep_codecommune
10 HAVING ventes >= 50
11 ORDER BY ventes desc
```

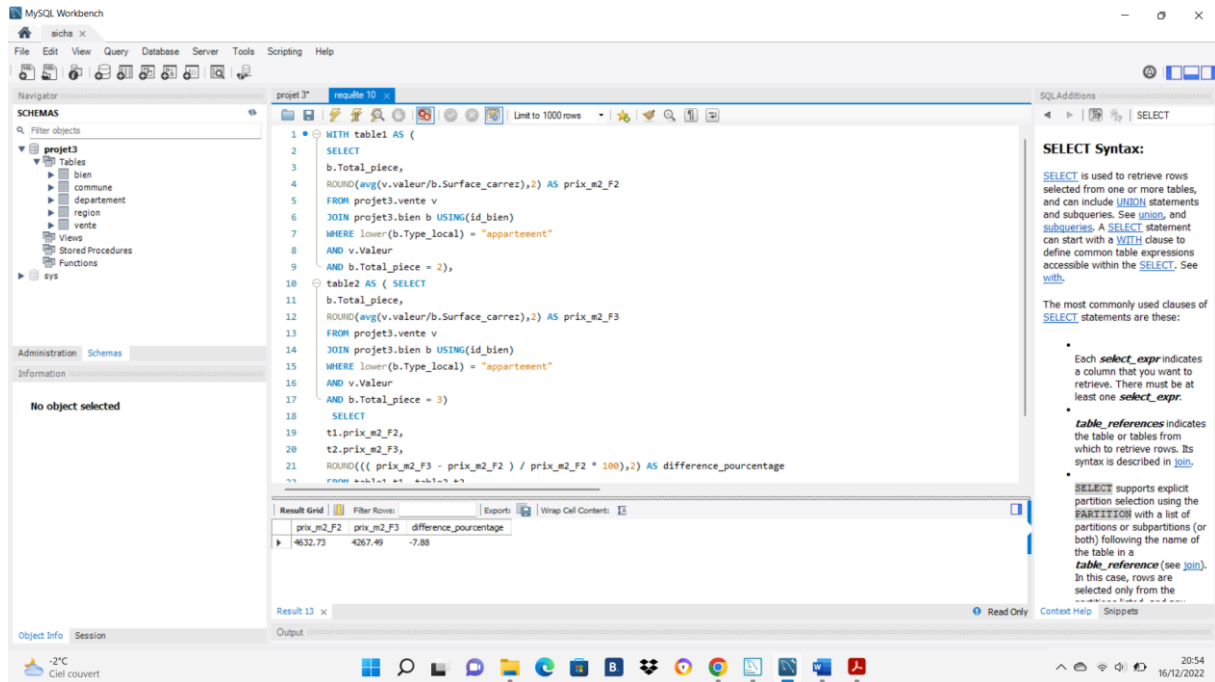
The 'Result Grid' at the bottom displays the results of the query, sorted by the number of sales in descending order. The columns are 'id_codedep_codecommune', 'Nom_commune', and 'ventes'.

id_codedep_codecommune	Nom_commune	ventes
075115	Paris 17e Arrondissement	228
075118	Paris 15e Arrondissement	215
006088	Nice	173
075111	Paris 11e Arrondissement	169
075116	Paris 16e Arrondissement	165
030663	Bordeaux	157
075114	Paris 14e Arrondissement	146
075120	Paris 20e Arrondissement	127
044109	Nantes	119
075119	Paris 19e Arrondissement	116
075112	Paris 12e Arrondissement	110
075110	Paris 10e Arrondissement	109
075109	Paris 9e Arrondissement	106
038185	Grenoble	106
092012	Orléans	99
075113	Paris 13e Arrondissement	94
075107	Paris 7e Arrondissement	87

The screenshot shows the MySQL Workbench interface with the same SQL query as the previous screenshot. The 'Result Grid' displays the continuation of the results, sorted by the number of sales in descending order.

id_codedep_codecommune	Nom_commune	ventes
075103	Paris 3e Arrondissement	79
031555	Toulouse	78
060604	Amboise	77
013204	Marseille 4e Arrondissement	72
013201	Marseille 1er Arrondissement	71
094080	Vincennes	68
092063	Rueil-Malmaison	68
059350	Lille	67
013209	Marseille 9e Arrondissement	66
093048	Montreuil	65
049007	Angers	64
030189	Nîmes	63
013028	La Ciotat	62
075108	Paris 8e Arrondissement	62
034301	Sète	62
035238	Reims	61
075102	Paris 2e Arrondissement	61
075104	Paris 4e Arrondissement	60
083137	Toulon	59
092044	Levallois-Perret	59
094068	Saint-Maur-des-Fossés	56
240004	Ajaccio	54
078646	Versailles	54
092062	Puteaux	53
092040	Issy-les-Moulineaux	50

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



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'project3' selected. The main editor window contains a SQL query for 'Requête 10'. The query uses CTEs to calculate the average price per square meter for 2-piece and 3-piece apartments, then calculates the percentage difference. The 'Result Grid' at the bottom shows one row of data.

```
1 WITH table1 AS (
2   SELECT
3     b.Total_piece,
4     ROUND(avg(v.valeur/b.Surface_carrez),2) AS prix_m2_F2
5   FROM project3.vente v
6   JOIN project3.bien b USING(id_bien)
7   WHERE lower(b.Type_local) = "appartement"
8   AND v.Valeur
9     AND b.Total_piece = 2),
10  table2 AS ( SELECT
11    b.Total_piece,
12    ROUND(avg(v.valeur/b.Surface_carrez),2) AS prix_m2_F3
13  FROM project3.vente v
14  JOIN project3.bien b USING(id_bien)
15  WHERE lower(b.Type_local) = "appartement"
16  AND v.Valeur
17    AND b.Total_piece = 3)
18  SELECT
19    t1.prix_m2_F2,
20    t2.prix_m2_F3,
21    ROUND((( prix_m2_F3 - prix_m2_F2 ) / prix_m2_F2 * 100),2) AS difference_pourcentage
22  FROM table1 t1, table2 t2
```

Result Grid:

prix_m2_F2	prix_m2_F3	difference_pourcentage
4632.73	4267.49	-7.88

SELECT Syntax:

SELECT is used to retrieve rows selected from one or more tables, and can include [JOIN](#) statements and subqueries. See [JOIN](#), and [subqueries](#). A **SELECT** statement can start with a **WITH** clause to define common table expressions accessible within the **SELECT**. See [with](#).

The most commonly used clauses of **SELECT** statements are these:

- Each **select_expr** indicates a column that you want to retrieve. There must be at least one **select_expr**.
- table_references** indicates the table or tables from which to retrieve rows. Its syntax is described in [join](#).
- SELECT** supports explicit partition selection using the **PARTITION** with a list of partitions or subpartitions (or both) following the name of the table in a **table_reference** (see [join](#)). In this case, rows are selected only from the

20:54
16/12/2022

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

MySQL Workbench

Navigation

SCHEMAS

Filter objects

project3

Tables

bien

commune

departement

region

vente

Views

Stored Procedures

Functions

sys

Administration

Schemas

Information

Table: bien

Columns:

id_bien int PK

no_vois varchar(10)

Type_vois varchar(10)

id_codedep_codecommune varchar(50)

Type_local varchar(10)

Surface_carrez float

Surface_local int

Total_piece int

project3 requete 11 requetes requete 12

Limit to 500 rows

```

3 WITH requete11 AS (select
4   b.id_codedep_codecommune,
5   c.nom_commune,
6   d.id_departement,
7   d.nom_departement,
8   ROUND(avg(v.valeur),2) AS "valeur_moyenne",
9   RANK()
10  OVER (PARTITION BY d.id_departement
11        ORDER BY ROUND(avg(v.valeur),2) desc) AS rang
12 FROM projet3.vente v
13 JOIN projet3.bien b USING(id_bien)
14 JOIN projet3.commune c USING(id_codedep_codecommune)
15 JOIN projet3.departement d USING(id_departement)
16 WHERE d.id_departement IN (006,013,033,059,069)
17 AND v.valeur
18 GROUP BY b.id_codedep_codecommune
19 )
20 SELECT
21   id_departement,
22   nom_departement,
23   id_codedep_codecommune,
24   nom_commune,
25   valeur_moyenne,
26   rang
27 FROM requete11
28 WHERE rang IN (1,2,3)

```

Result Grid

id_departement	nom_departement	id_codedep_codecommune	nom_commune	valeur_moyenne	rang
006	Alpes-Maritimes	006121	Saint-Jean-Cap-Ferrat	968750	1
006	Alpes-Maritimes	006059	Eze	655000	2
006	Alpes-Maritimes	006084	Mouans-Sartoux	476998.09	3
013	Bouches-du-Rhône	013043	Gignac-la-Nerthe	330000	1
013	Bouches-du-Rhône	013101	Saint-Savournin	314425	2

Result 3

Object Info

Session

10°C Nuageux

21:57 22/12/2022

MySQL Workbench

Navigation

SCHEMAS

Filter objects

project3

Tables

bien

commune

departement

region

vente

Views

Stored Procedures

Functions

sys

Administration

Schemas

Information

No object selected

project3 requete 11 requetes requete 12

Limit to 1000 rows

```

16 GROUP BY b.id_codedep_codecommune
17 )
18 SELECT
19   id_departement,
20   nom_departement,
21   id_codedep_codecommune,
22   nom_commune,
23   valeur_moyenne,
24   rang
25 FROM requete11
26 WHERE rang IN (1,2,3)
27

```

Result Grid

id_departement	nom_departement	id_codedep_codecommune	nom_commune	valeur_moyenne	rang
006	Alpes-Maritimes	006121	Saint-Jean-Cap-Ferrat	968750	1
006	Alpes-Maritimes	006059	Eze	655000	2
006	Alpes-Maritimes	006084	Mouans-Sartoux	476998.09	3
013	Bouches-du-Rhône	013043	Gignac-la-Nerthe	330000	1
013	Bouches-du-Rhône	013101	Saint-Savournin	314425	2
013	Bouches-du-Rhône	013022	Cassis	313416.88	3
033	Gironde	033236	Lège-Cap-Ferret	549500.64	1
033	Gironde	033539	Vayres	335000	2
033	Gironde	033059	Arcachon	307435.93	3
059	Nord	059071	Bersée	433202	1
059	Nord	059168	Cysong	408550	2
059	Nord	059279	Halluin	322250	3
069	Rhône	069265	Ville-sur-Jarniou	483300	1
069	Rhône	069382	Lyon 2e Arrondissement	455217.27	2
069	Rhône	069386	Lyon 6e Arrondissement	428968.25	3

Result 14

Object Info

Session

-2°C Ciel couvert

20:55 16/12/2022

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

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 WITH table_population AS (select
2   b.id_codedep_codecommune,
3   c.Nom_commune,
4   c.Population,
5   count(v.id_vente) AS vente
6 FROM projet3.vente v
7 JOIN projet3.bien b USING(id_bien)
8 JOIN projet3.commune c USING(id_codedep_codecommune)
9 WHERE c.Population > 10000
10 GROUP BY b.id_codedep_codecommune
11 )
12 SELECT
13   tp.id_codedep_codecommune,
14   tp.Nom_commune,
15   tp.vente,
16   tp.Population,
17   ROUND((tp.vente*1000/Population),2) AS vente_pour_1000habitants
18 FROM table_population tp
19 ORDER BY vente_pour_1000habitants DESC
20 LIMIT 20
```

The results are displayed in a table with the following columns: id_codedep_codecommune, Nom_commune, vente, Population, and vente_pour_1000habitants. The results are sorted in descending order of the calculated value.

id_codedep_codecommune	Nom_commune	vente	Population	vente_pour_1000habitants
075102	Paris 2e Arrondissement	127	21735	5.84
075101	Paris 1er Arrondissement	79	16055	4.92
075103	Paris 3e Arrondissement	161	34306	4.69
033009	Arcachon	55	11898	4.62
044055	La Baule-Escoubac	77	16797	4.58
075104	Paris 4e Arrondissement	120	29390	4.08
006104	Roquebrune-Cap-Martin	52	13041	3.99

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 WITH table_population AS (select
2   b.id_codedep_codecommune,
3   c.Nom_commune,
4   c.Population,
5   count(v.id_vente) AS vente
6 FROM projet3.vente v
7 JOIN projet3.bien b USING(id_bien)
```

The results are displayed in a table with the following columns: id_codedep_codecommune, Nom_commune, vente, Population, and vente_pour_1000habitants. The results are sorted in descending order of the calculated value.

id_codedep_codecommune	Nom_commune	vente	Population	vente_pour_1000habitants
075102	Paris 2e Arrondissement	127	21735	5.84
075101	Paris 1er Arrondissement	79	16055	4.92
075103	Paris 3e Arrondissement	161	34306	4.69
033009	Arcachon	55	11898	4.62
044055	La Baule-Escoubac	77	16797	4.58
075104	Paris 4e Arrondissement	120	29390	4.08
006104	Roquebrune-Cap-Martin	52	13041	3.99
075108	Paris 8e Arrondissement	139	36230	3.83
083123	Sanary-sur-Mer	60	17160	3.50
075109	Paris 9e Arrondissement	208	60563	3.43
083071	La Londe-les-Maures	27	10776	3.43
075106	Paris 6e Arrondissement	139	41171	3.38
083112	Saint-Cyr-sur-Mer	38	11725	3.24
060141	Chantilly	35	11178	3.13
044132	Pornichet	35	11440	3.06
094067	Saint-Mandé	69	22576	3.06
075110	Paris 10e Arrondissement	264	86863	3.04
006083	Menton	91	30981	2.94
085226	Saint-Hilaire-de-Riez	33	11501	2.87
094080	Vincennes	141	50230	2.81