

CRÉEZ ET UTILISEZ UNE BASE DE DONNÉES IMMOBILIÈRE  
AVEC SQL

# CRÉATION DE LA BASE DE DONNÉES

## 1.1 Dictionnaire de données

Les données utilisées étant issues de la base de données de Demandes de Valeurs Foncières du gouvernement, nous avons estimé qu'elles étaient préalablement vérifiées, **on ne les nettoie donc pas outre-mesure.**

Ensuite, on les élague pour ne **garder que les données utiles pour les requêtes** demandées.

1	id_bien	Identifiant du bien	Numérique	Identifiant
2	num_voie	Numéro de voie	Numérique	Smallint
3	compl_adresse	Complément d'adresse (B/T/Q)	Texte	Longueur : 3
4	type_voie	Type de voie (rue, avenue etc.)	Texte	Longueur : 10
5	voie	Nom de la voie	Texte	Longueur : 30
6	type_bien	Type du local (appartement, maison, etc.)	Texte	Longueur : 20
7	nb_pieces_principales	Nombre de pièces principales (hors cuisine, salles d'eau et dépendances)	Numérique	Smallint
8	nb_lots	Nombre de lots	Numérique	Smallint
9	surface_reelle_bati	Surface réelle du bâti (surface au sol entre les murs, arrondie au mètre carré inf.)	Numérique	Entier NOT NULL

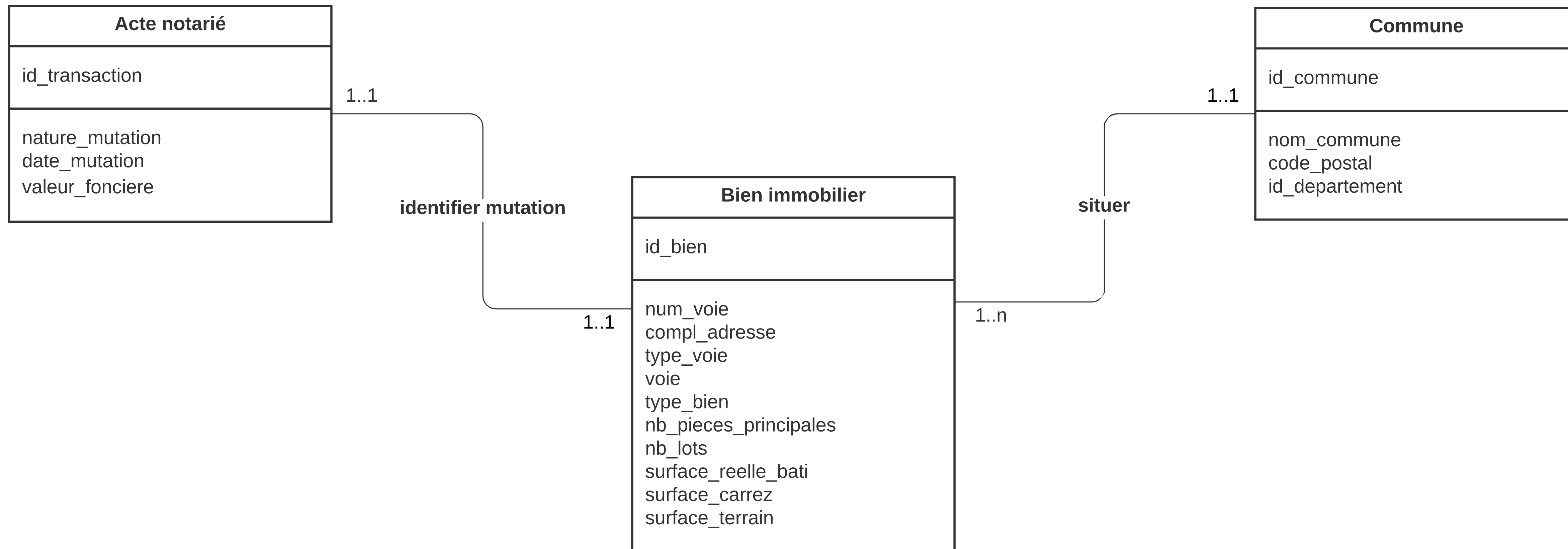
10	surface_carrez	Surface Carrez du premier lot	Numérique	Décimal NOT NULL
11	surface_terrain	Surface du terrain	Numérique	Décimal
12	id_transaction	Identifiant de la transaction	Numérique	Entier auto-incrémenté + Identifiant + [FK]
13	nature_mutation	Nature de la mutation	Texte	Longueur : 25
14	date_mutation	Date de mutation (signature de l'acte)	Date	AAAA-MM-JJ
15	valeur_fonciere	Valeur foncière du bien	Monétaire	Entier
16	id_commune	Identifiant de la commune	Numérique	Entier + Identifiant + [FK]
17	nom_commune	Nom de la commune	Texte	Longueur : 50
18	code_postal	Code postal	Numérique	Smallint
19	id_departement	Identifiant du département	Numérique	Smallint

## 1.2 Création du modèle conceptuel des données

Ces données peuvent être divisées en **3 tables** :

- ♦ Celle relative au **bien immobilier**
- ♦ Celle relative à l'**acte notarié**
- ♦ Celle relative à la **commune**

On utilise **Gitmind** pour rédiger le modèle conceptuel de données, les trois entités, les associations et cardinalités



## 1.3 Création du schéma relationnel normalisé en 3NF

Il faut également définir les clés primaires et étrangères

- ♦ Pour la table bien immobilier, on choisit un **identifiant unique auto-incrémenté**
- ♦ Pour acte notarié, on choisit un **identifiant unique auto-incrémenté**
- ♦ Pour commune, on choisit la **variable « code ID commune »**

Les variables ID commune et ID acte notarié seront également les clés étrangères de la table bien immobilier.

On utilise **SQL Power Architect** pour dessiner le schéma relationnel



New Project  
GestionTicket  
**P3-immo-OA**

Database

Acte\_notarie (TABLE)

Columns folder for Acte\_notarie

id\_transaction

nature\_mutation

date\_mutation

valeur\_fonciere

Foreign keys folder for Acte\_notarie

Foreign keys folder for Acte\_notarie

Commune (TABLE)

Columns folder for Commune

id\_commune

nom\_commune

code\_postal

id\_departement

Foreign keys folder for Commune

Foreign keys folder for Commune

Bien\_immobilier (TABLE)

Columns folder for Bien\_immobilier

id\_bien

num\_voie

compl\_adresse

type\_voie

voie

type\_bien

nb\_pieces\_principales

nb\_lots

surface\_reelle\_bati

surface\_carrez

surface\_terrain

id\_commune

id\_transaction

### Acte\_notarie

id\_transaction: INTEGER NOT NULL [ PK ]

nature\_mutation: VARCHAR(25) NOT NULL

date\_mutation: DATE NOT NULL

valeur\_fonciere: INTEGER NOT NULL

### Commune

id\_commune: INTEGER NOT NULL [ PK ]

nom\_commune: VARCHAR(50) NOT NULL

code\_postal: VARCHAR(5) NOT NULL

id\_departement: INTEGER NOT NULL

### Bien\_immobilier

id\_bien: INTEGER NOT NULL [ PK ]

num\_voie: INTEGER

compl\_adresse: VARCHAR(3)

type\_voie: VARCHAR(10) NOT NULL

voie: VARCHAR(30) NOT NULL

type\_bien: VARCHAR(20) NOT NULL

nb\_pieces\_principales: INTEGER NOT NULL

nb\_lots: INTEGER NOT NULL

surface\_reelle\_bati: INTEGER NOT NULL

surface\_carrez: FLOAT NOT NULL

surface\_terrain: FLOAT NOT NULL

id\_commune: INTEGER NOT NULL [ FK ]

id\_transaction: INTEGER NOT NULL [ FK ]





## 1.4 Import sur le Système de Gestion de Bases de Données

Sur SQL Power Architect, on exporte le code SQL issue de notre schéma relationnel.

Sur Postgres SQL, par PGAdmin, on créé une base données et on **importe le code SQL généré sur SQL Power Architect**.

Ensuite, on **importe les tables en csv**.

pgAdmin

FileObjectToolsHelp

Browser

FTS Configurations

FTS Dictionaries

AaFTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Procedures

1..3Sequences

Tables (3)

acte\_notarie

Columns (4)

id\_transaction

nature\_mutation

date\_mutation

valeur\_fonciere

Constraints

Indexes

RLS Policies

Rules

Triggers

bien\_immobilier

Columns

Constraints

Indexes

RLS Policies

DashboardPropertiesSQLStatisticsDependenciesDependentsP3/postgres@...public.acte\_not...publi<>Xri

Query EditorQuery HistoryExplainMessagesNotifications

Successfully run. Total query runtime: 256 msec.  
34169 rows affected.

Data Output

	id_transaction [PK] integer	nature_mutation character varying (25)	date_mutation date	valeur_fonciere integer
1	1	Vente	2020-02-03	56000
2	2	Vente	2020-01-02	165000
3	3	Vente	2020-01-08	720000
4	4	Vente	2020-01-06	429250
5	5	Vente	2020-01-07	220900
6	6	Vente	2020-01-21	42000
7	7	Vente	2020-01-07	262000
8	8	Vente	2020-01-08	190000
9	9	Vente	2020-01-16	563130
10	10	Vente	2020-01-17	535000
11	11	Vente	2020-01-16	330000
12	12	Vente	2020-01-27	110600
13	13	Vente	2020-01-30	50000
14	14	Vente	2020-01-09	212000
15	15	Vente	2020-01-15	160000

Successfully run. Total query runtime: 334 msec.  
34169 rows affected.

### Data Output

	 id_bien [PK] integer	 num_voie integer	 compl_adresse character varying (3)	 type_voie character varying (10)	 voie character varying (30)	 type_bien character varying (20)	 nb_pieces_pri integer
1	1	190	A	RUE	CENTRALE	Appartement	
2	2	347	[null]	RUE	DU CHATEAU	Appartement	
3	3	58	[null]	AV	DU MONT BLANC	Appartement	
4	4	140	[null]	RUE	DE L'ABBE JOLIVET	Maison	
5	5	39	[null]	RUE	BUFFON	Appartement	
6	6	28	[null]	AV	JEAN FALCONNIER	Appartement	
7	7	8	[null]	RUE	DE GENEVE	Appartement	
8	8	2	[null]	RUE	DU RECULET	Appartement	
9	9	1403	[null]	RUE	JEAN DE GINGINS	Maison	
10	10	226	[null]	ALL	DES CAPUCINES	Maison	
11	11	276	[null]	RTE	DE POUIGNY	Appartement	
12	12	79	[null]	CRS	DE VERDUN	Appartement	
13	13	77	B	RUE	DU COMMERCE	Appartement	
14	14	240	[null]	RUE	DE PRE BAILLY	Appartement	
15	15	3	[null]	RUE	TURENNE	Appartement	

Successfully run. Total query runtime: 235 msec.  
3215 rows affected.

### Data Output

	id_commune [PK] integer	nom_commune character varying (50)	code_postal character varying (5)	id_departement character varying
1	0	SAINT-ETIENNE-DU-BOIS	1370	1
2	1	CHEVRY	1170	1
3	2	DIVONNE-LES-BAINS	1220	1
4	3	PERON	1630	1
5	4	VALSERHONE	1200	1
6	5	CULOZ	1350	1
7	6	ST-GENIS-POUILLY	1630	1
8	7	OYONNAX	1100	1
9	8	ST-GERMAIN-DE-JOUX	1130	1
10	9	GEX	1170	1
11	10	AMBERIEU-EN-BUGEY	1500	1
12	11	CESSY	1170	1
13	12	SAULT-BRENAZ	1150	1
14	13	FERNEY-VOLTAIRE	1210	1
15	14	SEGNY	1170	1
16	15	LAGNY	1150	1

# REQUÊTES SQL



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

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Browser' pane displays the database structure for 'P3', including 'Databases (3)', 'P3', 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas (1)', 'public', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', and 'Tables (3)'. The 'Tables (3)' section is expanded, showing 'acte\_notarie' and 'bien\_immobilier'.

The main pane shows the 'Query Editor' with the following SQL query:

```
1 SELECT COUNT (DISTINCT id_bien) AS Nombre_appartements_vendus_1er_semestre_2020
2 FROM bien_immobilier
3 INNER JOIN acte_notarie ON acte_notarie.id_transaction = bien_immobilier.id_transaction
4 WHERE type_bien = 'Appartement' AND date_mutation BETWEEN '2020-01-01' AND '2020-06-30';
```

Below the query editor, the 'Data Output' pane shows the result of the query:

	nombre_appartements_vendus_1er_semestre_2020
	bigint
1	31378

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

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to show the 'public' schema. The 'Query Editor' tab is active, displaying a SQL query. Below the query, the 'Data Output' tab shows the results of the query in a table format.

**Query:**

```
1 SELECT nb_pieces_principales,  
2 ROUND (COUNT (nb_pieces_principales) * 100.0 / (SELECT COUNT (*) FROM bien_immobilier), 3)  
3 AS proportion_des_ventes  
4 FROM bien_immobilier  
5 WHERE type_bien = 'Appartement'  
6 GROUP BY nb_pieces_principales  
7 ORDER BY proportion_des_ventes DESC;
```

**Data Output:**

	nb_pieces_principales integer	proportion_des_ventes numeric
1	2	28.631
2	3	26.240
3	1	19.723
4	4	13.053
5	5	3.260
6	6	0.597
7	7	0.158
8	0	0.088
9	8	0.050
10	9	0.023
11	10	0.006
12	11	0.003

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

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to show the 'public' schema. The 'Query Editor' tab is active, displaying a SQL query that calculates the average price per square meter for each department. The 'Data Output' tab shows the results of the query, which are sorted by price per square meter in descending order, limited to the top 10 results.

**Query Editor**

```
1 SELECT id_departement, ROUND (AVG (valeur_fonciere / surface_carrez) ) AS prix_m2
2 FROM acte_notarie
3 INNER JOIN bien_immobilier ON bien_immobilier.id_transaction = acte_notarie.id_transaction
4 INNER JOIN commune ON commune.id_commune = bien_immobilier.id_commune
5 GROUP BY id_departement
6 ORDER BY prix_m2 DESC
7 LIMIT 10;
```

**Data Output**

	id_departement	prix_m2
	character varying	double precision
1	75	12045
2	92	7219
3	94	5341
4	6	4697
5	74	4667
6	93	4337
7	78	4225
8	69	4059
9	2A	4011
10	33	3764



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

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to show the 'public' schema, with the 'acte\_notarie' table selected. The main panel displays a SQL query in the 'Query Editor' tab. The query calculates the average price per square meter for houses in Île-de-France. Below the query, the 'Data Output' tab shows the result of the query, which is a single row with the value 3745.

**Query Editor**

```
1 SELECT ROUND (AVG (valeur_fonciere / surface_carrez) ) AS prix_m2_moyen
2 FROM bien_immobilier
3 INNER JOIN acte_notarie ON acte_notarie.id_transaction = bien_immobilier.id_transaction
4 INNER JOIN commune ON commune.id_commune = bien_immobilier.id_commune
5 WHERE type_bien = 'Maison' AND id_departement IN ('75', '77', '78', '91', '92', '93', '94', '95');
```

**Data Output**

	prix_m2_moyen double precision
1	3745

5. Liste des 10 appartements les plus chers avec le département et le nombre de mètres carrés.

pgAdmin

FileObjectToolsHelp

Browserservers (1)P3Databases (3)P3CastsCatalogsEvent TriggersExtensionsForeign Data WrappersLanguagesPublicationsSchemas (1)publicCollationsDomainsFTS ConfigurationsFTS DictionariesFTS ParsersFTS TemplatesForeign TablesFunctionsMaterialized ViewsProceduresSequencesTables (3)acte\_notariebien immobilier

DashboardPropertiesSQLStatisticsDependenciesDependentsP3/postgres@...P3/postgres@...P3/post<>x

Query EditorQuery HistoryExplainNotifications

```
1 SELECT valeur_fonciere, id_departement, surface_carrez
2 FROM bien_immobilier
3 INNER JOIN commune ON commune.id_commune = bien_immobilier.id_commune
4 INNER JOIN acte_notarie ON acte_notarie.id_transaction = bien_immobilier.id_transaction
5 WHERE type_bien = 'Appartement'
6 ORDER BY valeur_fonciere DESC
7 LIMIT 10;
```

Data OutputMessages

	valeur_fonciere integer	id_departement character varying	surface_carrez real
1	9000000	75	9.1
2	8600000	91	64
3	8577713	75	20.55
4	7620000	75	42.77
5	7600000	75	253.3
6	7535000	75	139.9
7	7420000	75	360.95
8	7200000	75	595
9	7050000	75	122.56
10	6600000	75	79.38

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

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to show the 'public' schema. The 'Query Editor' tab is active, displaying a SQL query that calculates the percentage change in sales between the first and second quarters of 2020. The query uses CTEs to define the sales for each quarter and then calculates the percentage change using the ROUND function.

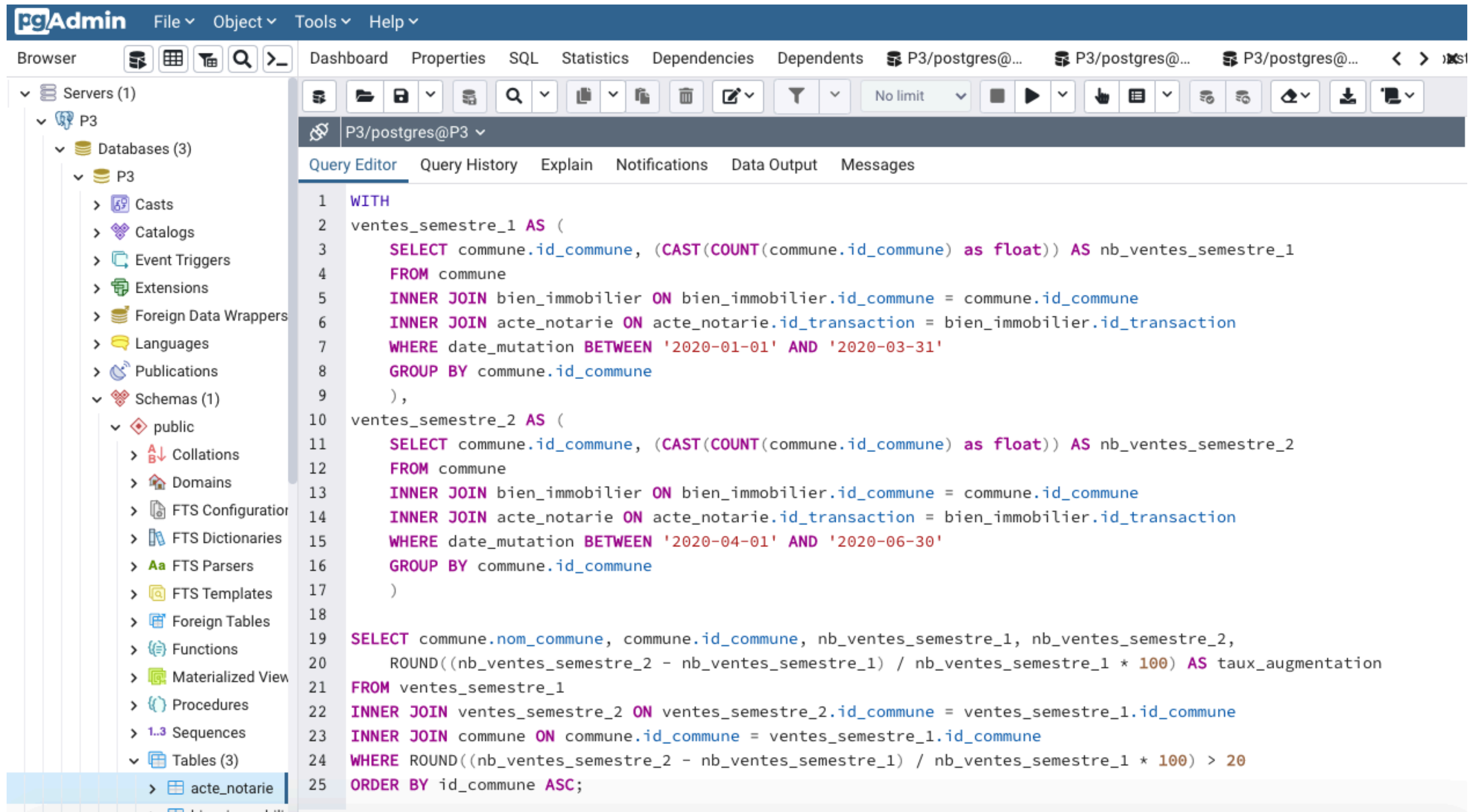
```
1 WITH
2 ventes_trimestre_1 AS (
3     SELECT COUNT (id_transaction) AS nb_ventes_trimestre_1
4     FROM acte_notarie
5     WHERE date_mutation BETWEEN '2020-01-01' AND '2020-03-31'
6 ),
7 ventes_trimestre_2 AS (
8     SELECT COUNT (id_transaction) AS nb_ventes_trimestre_2
9     FROM acte_notarie
10    WHERE date_mutation BETWEEN '2020-04-01' AND '2020-06-30'
11 )
12 SELECT ROUND (((nb_ventes_trimestre_2 - nb_ventes_trimestre_1) :: numeric / nb_ventes_trimestre_1 * 100), 2)
13 AS taux
14 FROM ventes_trimestre_1, ventes_trimestre_2;
```

The 'Data Output' tab shows the result of the query, which is a single row with the value 3.68.

taux
3.68



7. Communes où le nombre de ventes a augmenté d'au moins 20% entre le 1er et le 2nd trimestre.



The screenshot displays the pgAdmin 4 web interface. On the left, the 'Servers' tree shows a connection to 'P3', which contains a database 'P3'. Under 'P3', there is a 'public' schema containing several objects, including 'acte\_notarie'. The main area is the 'Query Editor', which is active and shows a SQL query. The query is designed to find communes where the number of sales increased by at least 20% from the first quarter to the second quarter of 2020. It uses Common Table Expressions (CTEs) to calculate the number of sales for each commune in each quarter, then joins these results to calculate the percentage increase.

```
1 WITH
2 ventes_semestre_1 AS (
3     SELECT commune.id_commune, (CAST(COUNT(commune.id_commune) as float)) AS nb_ventes_semestre_1
4     FROM commune
5     INNER JOIN bien_immobilier ON bien_immobilier.id_commune = commune.id_commune
6     INNER JOIN acte_notarie ON acte_notarie.id_transaction = bien_immobilier.id_transaction
7     WHERE date_mutation BETWEEN '2020-01-01' AND '2020-03-31'
8     GROUP BY commune.id_commune
9 ),
10 ventes_semestre_2 AS (
11     SELECT commune.id_commune, (CAST(COUNT(commune.id_commune) as float)) AS nb_ventes_semestre_2
12     FROM commune
13     INNER JOIN bien_immobilier ON bien_immobilier.id_commune = commune.id_commune
14     INNER JOIN acte_notarie ON acte_notarie.id_transaction = bien_immobilier.id_transaction
15     WHERE date_mutation BETWEEN '2020-04-01' AND '2020-06-30'
16     GROUP BY commune.id_commune
17 )
18
19 SELECT commune.nom_commune, commune.id_commune, nb_ventes_semestre_1, nb_ventes_semestre_2,
20     ROUND((nb_ventes_semestre_2 - nb_ventes_semestre_1) / nb_ventes_semestre_1 * 100) AS taux_augmentation
21 FROM ventes_semestre_1
22 INNER JOIN ventes_semestre_2 ON ventes_semestre_2.id_commune = ventes_semestre_1.id_commune
23 INNER JOIN commune ON commune.id_commune = ventes_semestre_1.id_commune
24 WHERE ROUND((nb_ventes_semestre_2 - nb_ventes_semestre_1) / nb_ventes_semestre_1 * 100) > 20
25 ORDER BY id_commune ASC;
```



P3/postgres@P3 ▾

Query Editor

Query History

Explain

Notifications

Data Output

Messages

	<b>nom_commune</b> character varying (50)	<b>id_commune</b> [PK] integer	<b>nb_ventes_semestre_1</b> double precision	<b>nb_ventes_semestre_2</b> double precision	<b>taux_augmentation</b> double precision
1	LAON	55	11	14	27
2	VILLERS-COTTERETS	65	3	5	67
3	CHATEAU-ARNOUX-SAINT-AUBAN	80	1	2	100
4	BARCELONNETTE	90	2	5	150
5	SAINT-MARTIN-DE-BROMES	92	1	2	100
6	EMBRUN	114	1	2	100
7	ORCIERES	117	1	5	400
8	GAP	120	2	6	200
9	LE DEVOLUY	121	1	11	1000
10	LA SALLE	124	1	3	200
11	RISOUL	129	1	2	100
12	NICE	144	23	32	39
13	NICE	148	33	61	85
14	MENTON	149	40	51	28
15	SAINT-MARTIN-VESUBIE	172	2	6	200
16	GILETTE	181	1	2	100
17	LEVENS	185	1	4	300
18	BEUIL	186	1	2	100
19	ANTIBES	205	24	29	21
20	VALBONNE	222	1	2	100



P3/postgres@P3 ▾

[Query Editor](#)[Query History](#)[Explain](#)[Notifications](#)[Data Output](#)[Messages](#)

	<b>nom_commune</b> character varying (50)	<b>id_commune</b> [PK] integer	<b>nb_ventes_semestre_1</b> double precision	<b>nb_ventes_semestre_2</b> double precision	<b>taux_augmentation</b> double precision
573	BEZONS	3157	8	18	125
574	MARIN	3163	2	4	100
575	FORT DE FRANCE	3164	9	11	22
576	TROIS ILETS	3165	6	8	33
577	DUCOS	3167	3	10	233
578	LE LAMENTIN	3168	6	8	33
579	SAINTE LUCE	3172	1	3	200
580	MATOURY	3182	1	2	100
581	SAINT DENIS	3186	5	8	60
582	SAINT DENIS	3187	6	10	67
583	SAINT-PAUL	3189	1	3	200
584	PARIS 01	3196	34	45	32
585	PARIS 08	3197	62	77	24
586	PARIS 10	3201	109	155	42
587	PARIS 11	3203	169	214	27
588	PARIS 12	3204	110	144	31
589	PARIS 16	3208	165	229	39
590	PARIS 18	3210	209	307	47
591	PARIS 20	3211	127	176	39
592	PARIS 19	3212	116	151	30



## 8. Différence (en %) du prix au mètre carré entre les appartements de 2 et 3 pièces

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Servers' tree is expanded to show the 'public' schema. The 'Query Editor' tab is active, displaying a SQL query that calculates the percentage difference in price per square meter between 2-room and 3-room apartments. The query uses CTEs to calculate the average price per square meter for each room type and then computes the difference.

```
1 WITH
2 prix_2_pieces AS (
3     SELECT AVG (valeur_fonciere / surface_carrez) AS prix_m2_2pieces
4     FROM acte_notarie
5     INNER JOIN bien_immobilier ON bien_immobilier.id_transaction = acte_notarie.id_transaction
6     WHERE type_bien = 'Appartement' AND nb_pieces_principales = '2'
7 ),
8 prix_3_pieces AS (
9     SELECT AVG (valeur_fonciere / surface_carrez) AS prix_m2_3pieces
10    FROM acte_notarie
11    INNER JOIN bien_immobilier ON bien_immobilier.id_transaction = acte_notarie.id_transaction
12    WHERE type_bien = 'Appartement' AND nb_pieces_principales = '3'
13 )
14 SELECT ROUND (((prix_m2_3pieces - prix_m2_2pieces) / prix_m2_2pieces * 100) :: numeric, 2) AS difference_prix
15 FROM prix_2_pieces, prix_3_pieces;
```

The 'Data Output' section shows the result of the query:

	difference_prix numeric
1	-12.31

9. Valeurs foncière moyenne pour le top 3 des communes des départements 6, 13, 33, 59 et 69.

pgAdmin

File Object Tools Help

Browsers

Dashboard Properties SQL Statistics Dependencies Dependents P3/postgres@... P3/postgres@... P3/postgres@... < > >st

Servers (1)

- P3
  - Databases (3)
    - P3
      - Casts
      - Catalogs
      - Event Triggers
      - Extensions
      - Foreign Data Wrappers
      - Languages
      - Publications
      - Schemas (1)
        - public
          - Collations
          - Domains
          - FTS Configuration
          - FTS Dictionaries
          - FTS Parsers
          - FTS Templates
          - Foreign Tables
          - Functions
          - Materialized View
          - Procedures
          - Sequences
          - Tables (3)
            - acte\_notarie
            - bien\_immobilier

P3/postgres@P3

Query Editor Query History Explain Notifications

1 WITH

2 moyenne\_ville AS (

3     SELECT

4         ROW\_NUMBER() OVER (

5             PARTITION BY id\_departement

6             ORDER BY AVG(valeur\_fonciere) DESC) AS top\_communes,

7         id\_departement, nom\_commune, AVG(valeur\_fonciere) AS valeur\_moyenne

8     FROM bien\_immobilier

9     INNER JOIN commune ON commune.id\_commune = bien\_immobilier.id\_commune

10    INNER JOIN acte\_notarie ON acte\_notarie.id\_transaction = bien\_immobilier.id\_transaction

11    WHERE id\_departement IN ('6', '13', '33', '59', '69')

12    GROUP BY id\_departement, nom\_commune

13    )

14

15    SELECT (id\_departement) :: int, nom\_commune, ROUND(valeur\_moyenne, 2) AS valeur\_moyenne, top\_communes

16    FROM moyenne\_ville

17    WHERE top\_communes <= 3

18    ORDER BY id\_departement ASC, top\_communes ASC;

Data Output Messages

	id_departement integer	nom_commune character varying (50)	valeur_moyenne numeric	top_communes bigint
1	6	SAINT-JEAN-CAP-FERRAT	968750.00	1
2	6	EZE	655000.00	2
3	6	MOULANS CAPTOLUX	476000.00	3





P3/postgres@P3 ▾

Query Editor

Query History

Explain

Notifications

Data Output

Messages

	<b>id_departement</b> integer	<b>nom_commune</b> character varying (50)	<b>valeur_moyenne</b> numeric	<b>top_communes</b> bigint
1	6	SAINT-JEAN-CAP-FERRAT	968750.00	1
2	6	EZE	655000.00	2
3	6	MOUANS-SARTOUX	476898.13	3
4	13	GIGNAC-LA-NERTHE	330000.00	1
5	13	SAINT SAVOURNIN	314425.00	2
6	13	CASSIS	313416.88	3
7	33	LEGE-CAP-FERRET	549500.64	1
8	33	VAYRES	335000.00	2
9	33	ARCACHON	307435.93	3
10	59	BERSEE	433202.00	1
11	59	CYSOING	408550.00	2
12	59	HALLUIN	322250.00	3
13	69	VILLE SUR JARNIOUX	485300.00	1
14	69	LYON 2EME	455217.26	2
15	69	LYON 6EME	426968.25	3