



Data-Immo

Les données au cœur de l'immobilier

Sommaire

Les fondations de la base de données

Dictionnaire des données	p.4
Définition des tables	p.5
Schéma relationnel	p.6
Configuration des tables	p.7
Schéma conceptuel	p.8

Création de la base de données

Création du script	p.13
Chargement de la base de données	p.14
Base de données opérationnelle	p.15

Préparation des données

Nettoyage des données	p.10
Création du fichier des tables	p.11

Requêtes et résultats

Requêtes et résultats	p.17 à 27
-----------------------	-----------

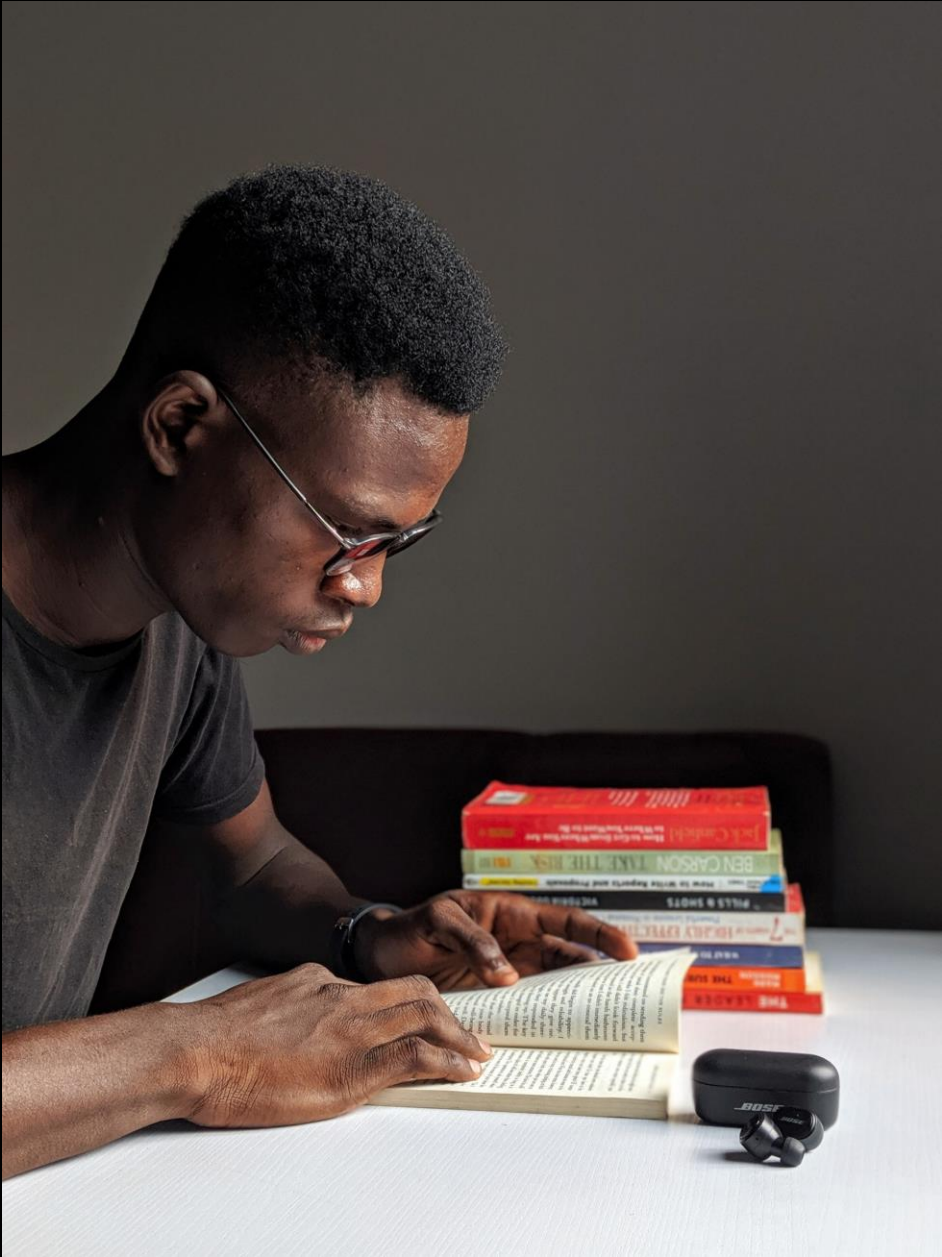
Annexes

Dictionnaire des données	suite présentation
Script SQL de la base de données	suite présentation
Résultats complet des requêtes	fichier .xls externe





Les fondations de la base de données



Dictionnaire des données

Afin de faciliter des modifications et la reprise d'informations par un tiers, toutes les informations de la base sont répertoriées dans un répertoire dédié.

(Dictionnaire complet en annexe p.29)

numero	code propriete	signification	type	observation
9	Date mutation	Date de vente / cession	Date	format ISO-8601 (YYYY-MM-DD)
18	Code ID commune	Code commune INSEE	Numerique	Longueur : 5 caracteres
20	Commune	Nom de la commune	Alphanumerique	Composition lettres et chiffres
43	Nature culture	Type de culture du terrain	Texte	Vegetation presente sur le terrain

Définition des tables

“

Que contiendront les tables ?



bien

La table « bien » donnera les caractéristiques technique du ou des biens. Cette dernière permettra d'identifier chaque bien de manière unique malgré les similitudes avec d'autres.

vente

La table « vente » contiendra des informations liées aux transactions des biens, tel que le prix, la nature ou encore la date.

cadastre

La table « cadastre » fournira des informations sur l'environnement de chaque bien mais aussi des caractéristiques liées aux déclarations administratives des biens. Cette partie de la base de données sera pertinente seulement pour des besoins spécifiques ou des recherches plus approfondies.

adresse

La table « adresse » correspondra aux informations de localisation du ou des biens, tel que la rue, la commune, le département ...

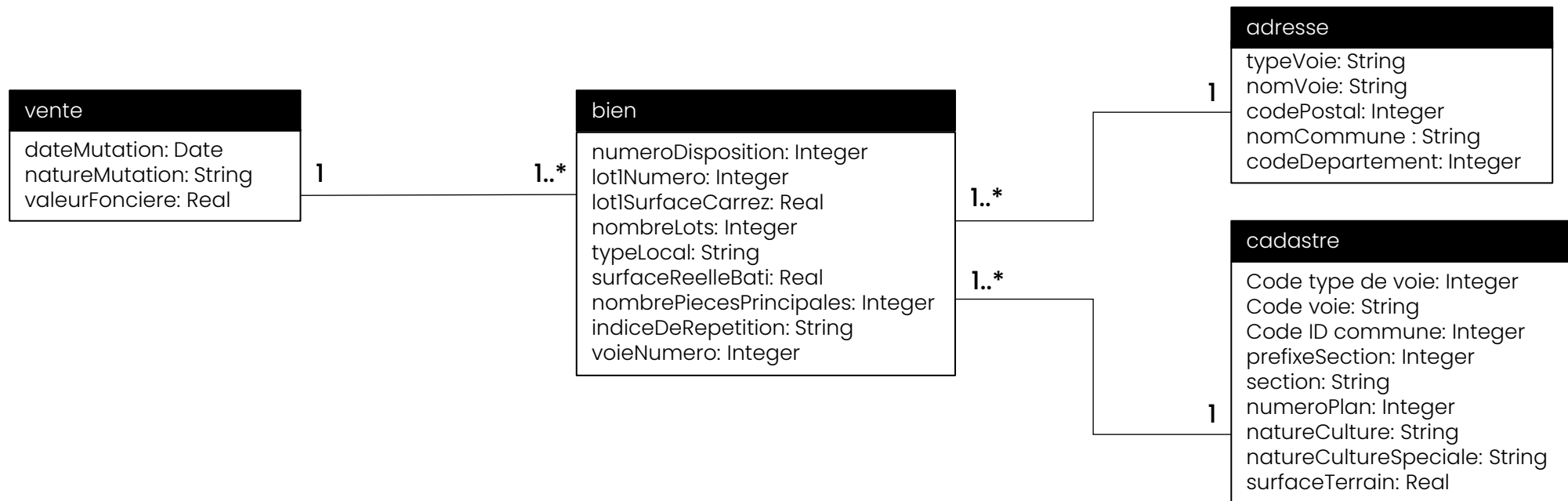
Schéma relationnel

“

La structure de la base de données est représenté par un schéma en langage UML.

Ce dernier met en évidence les différentes tables avec leurs relations ainsi que leurs cardinalités.

Certaines contraintes tel que la norme 3NF consistant à ce qu'il n'y ai pas de dépendance entre attributs clés et non clés, sauf exception, doivent être respectées.



Configuration des tables

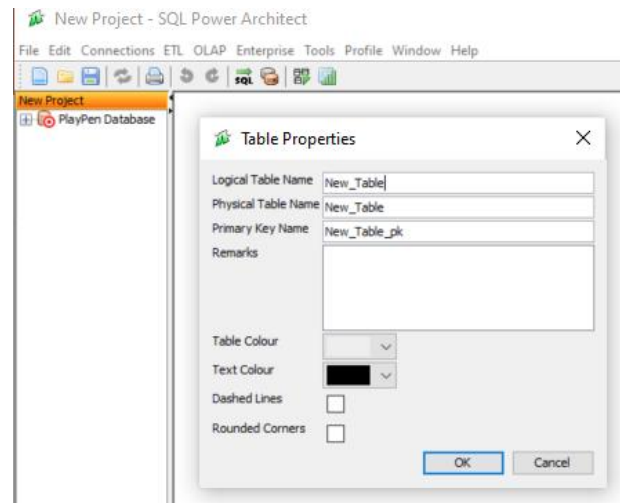
“

La structure de la base de donnée

Création des tables

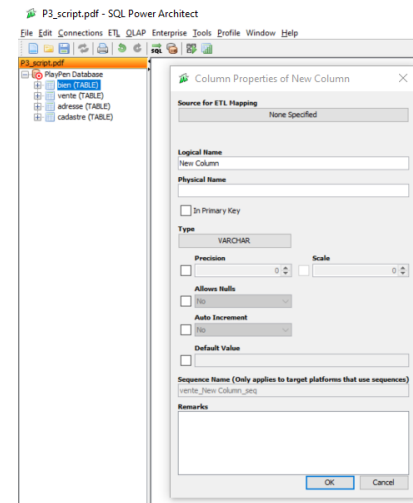


La première étape consiste à créer l'ensemble des tables et de les nommer sur un logiciel spécialisé..



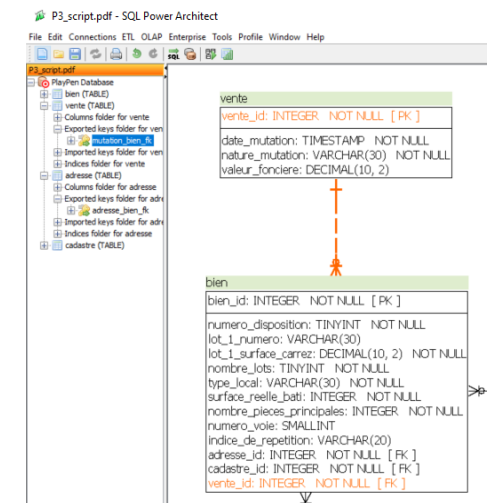
Création des attributs

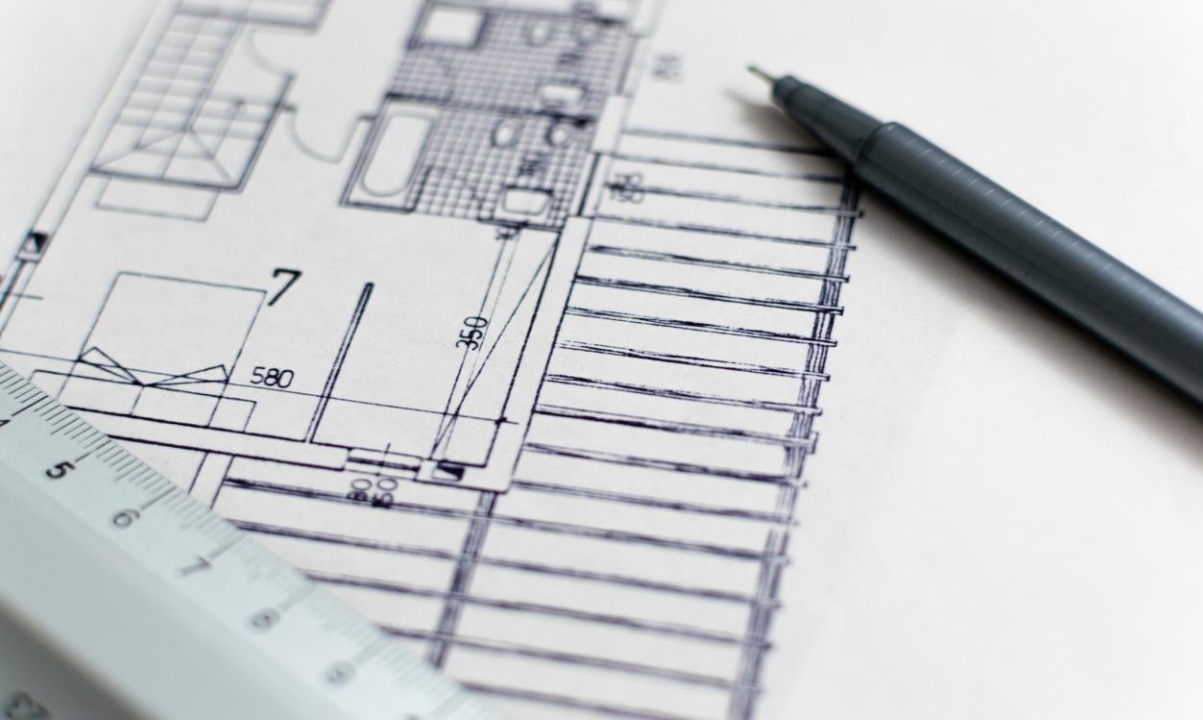
La seconde étape consiste à créer l'ensemble des attributs de chaque table, en définissant pour chacun le type, la taille et leur fonction (PK, NOT NULL, VARCHAR ...).



Mise en place des relations

La 3^e étape consiste à lier les tables entre elles en fonction du schéma relationnel établie via des relations identifiantes ou non.

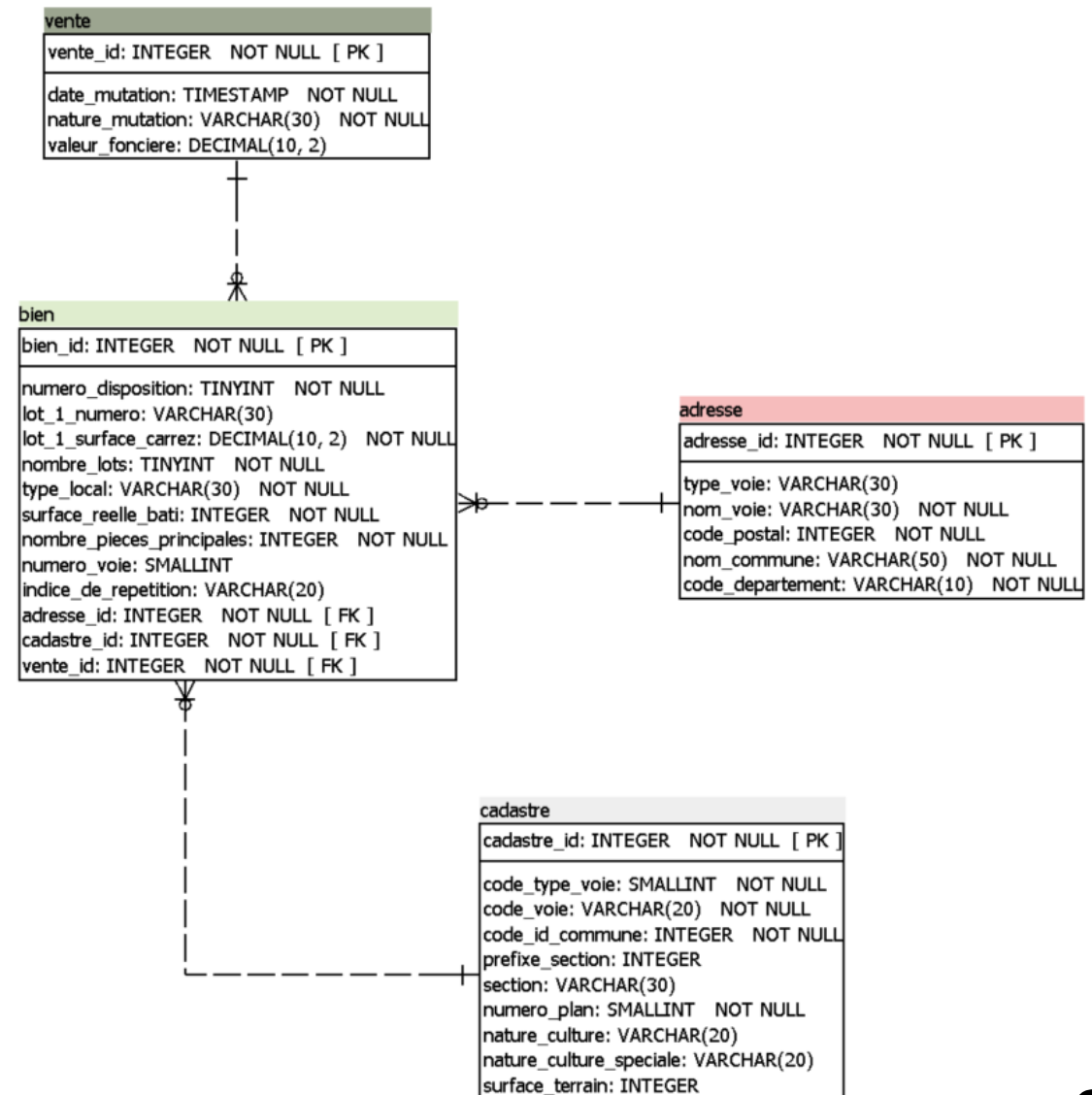




Modèle conceptuel des données

Ce modèle représente l'ensemble des tables de la base de données avec leurs attributs, leurs clés ainsi que leurs relations.

(Script SQL en annexe p.30)





Préparation des données

Nettoyage des données

“

Conserver des données pertinentes

Nettoyage des données

Le nettoyage des données permet de trier et conserver les données de façon pertinente. Dans ce fichier, j'ai volontairement écarté certaines colonnes contenant des données nulles ou non représentatives (-1%).

The screenshot shows an Excel spreadsheet with the following columns: Articles CGI 4, Articles CGI 5, Articles CGI No disposition, Date mutation, Nature mutation, Valeur foncière, No voie, B/T/Q, Code type de voie, Type de voie, Code voie, Voie, Code ID commune, and Code postal. The data rows show various transactions with dates ranging from 2020-01-01 to 2020-01-31, including details on the nature of the mutation (e.g., Vente), the value, and the address.

Manipulations et adaptation des données

A l'aide de l'éditeur Power Query, le fichier est ensuite traité, chaque colonne conservera ou non le type de données fourni par défaut et subira des modifications le cas échéant.

The screenshot shows the Power Query Editor interface with a table containing the same data as the Excel spreadsheet. The table has columns for articles, dates, nature of mutation, value, and address. The interface includes various options for transforming and filtering the data, such as 'Filtrer', 'Transformer', and 'Fusionner des requêtes'.

Création du fichier des tables

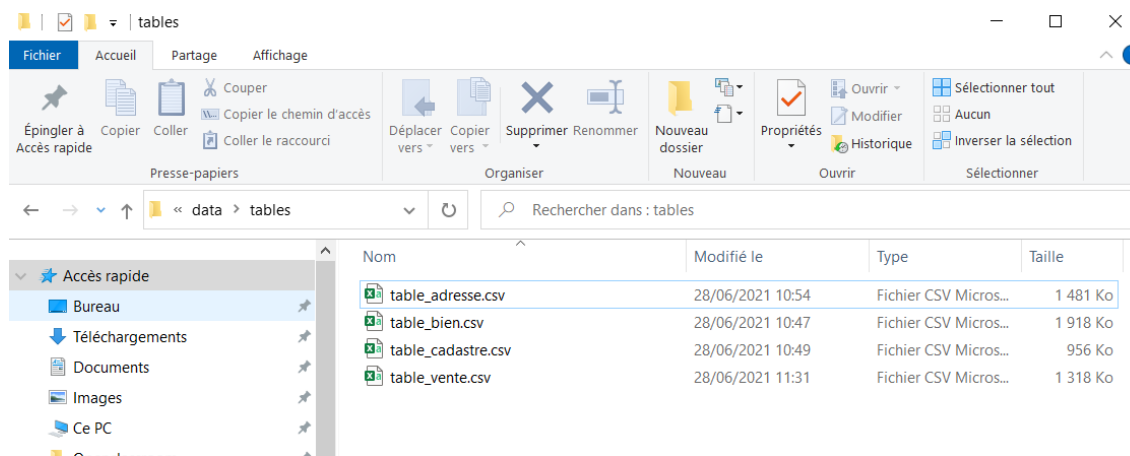


Préparation des tables

Division des données en tables

Une fois le fichier nettoyé et les colonnes configurées, le fichier sera scinder en plusieurs tables, suivant le schéma relationnel qui prendront chacune la forme d'un fichier CSV.

NB : Les clés primaires et étrangères (PK, FK) seront intégrées manuellement car les données sont déjà existantes.



	A	B	C	D	E	F	G
	bien_id	numero_dispo	lot_1_numero	lot_1_surface	nombre_lots	type_local	surface_reelle
1	1	1	22	50,42	1	Appartement	52
2	2	1	12	48,22	2	Appartement	48
3	3	1	146	130,8	2	Appartement	130
4	4	1	11	109,22	1	Maison	109
5	5	1	31	108,65	2	Appartement	91
6	6	1	50	31,65	1	Appartement	32
7	7	1	11	52,58	1	Appartement	52
8	8	1	304	58,71	1	Appartement	60
9	9	1	14	93,23	3	Maison	96
10	10	1	21	117	1	Maison	117
11	11	1	4	35,6	1	Appartement	36
12	12	1	169	138,03	2	Appartement	137
13	13	1	11	42	1	Appartement	43
14	14	1	110	45,36	2	Appartement	45
15	15	1	213	82,6	2	Appartement	84
16	16	1	9	88,11	1	Appartement	88
17	17	1	3	164,87	1	Appartement	143
18	18	1	47	42,2	2	Appartement	42
19	19	1	4	68,11	1	Appartement	65
20	20	1	1	82,59	1	Appartement	75
21	21	1	9	30,23	1	Appartement	30
22	22	1	2	24,8	1	Appartement	25
23	23	1	60	52,6	1	Appartement	52
24	24	1	4	91,69	1	Appartement	88
25	25	1	1	45,54	1	Appartement	45
26	26	1	41	84,24	1	Appartement	83
27	27	1	18	68,23	1	Appartement	68
28	28	1	2	62,03	1	Appartement	64
29	29	1	74	112,93	1	Maison	109



Création de la base de données

Création du script

“

Le script de la base de données

Création du code des tables

Cette étape consiste à créer la structure de la base de donnée dans le SGBD.

Exemple avec la table VENTE :

-- création de la séquence qui permettra d'incrémenter une série pour la clé primaire (PK)

```
CREATE SEQUENCE public.vente_vente_id_seq;
```

-- création de la table VENTE avec ses attributs

```
CREATE TABLE public.vente
```

-- création de la clé primaire (PK) avec son type INTEGER (nombre entier), un e caractéristique NOT NULL (qui doit obligatoirement contenir une valeur) et une valeur suivante par défaut suivant la séquence.

```
(vente_id INTEGER NOT NULL DEFAULT  
nextval('public.vente_vente_id_seq'),
```

-- ajout des autres attributs

```
date_mutation TIMESTAMP NOT NULL,  
nature_mutation VARCHAR(30) NOT NULL,  
valeur_foncieres NUMERIC(10,2),
```

-- Ajout d'un attribut avec contrainte pour la clé étrangère

```
CONSTRAINT vente_pk PRIMARY KEY (vente_id)  
);  
ALTER SEQUENCE public.vente_vente_id_seq OWNED BY  
public.vente.vente_id;
```

Création du code des clés étrangères

Cette étape consiste à créer les clés qui permettront de relier les tables entre elle et d'indiquer l'action qui sera réalisée en cas de modification de ces dernières.

NB : Les attributs de ces clés ont déjà été incluses dans les tables

Exemple :

-- modification de la table BIEN pour ajouter la contrainte de clé étrangère (FK) cadastre.bien_fk

```
ALTER TABLE public.bien ADD CONSTRAINT  
cadastre_bien_fk
```

-- nom de la clé étrangère

```
FOREIGN KEY (cadastre_id)
```

-- provenance de la clé étrangère

```
REFERENCES public.cadastre (cadastre_id)
```

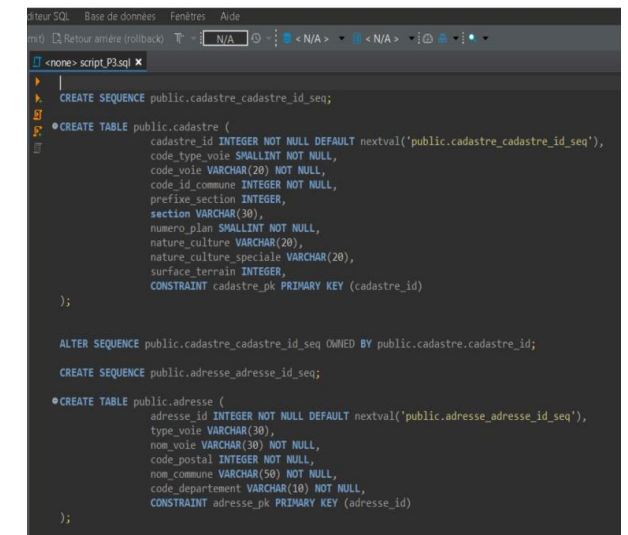
-- Ce bloc consiste à spécifier les actions à suivre en cas de suppression ou modification des données.

```
ON DELETE NO ACTION  
ON UPDATE NO ACTION  
NOT DEFERRABLE;
```

Chargement et exécution du code

Une fois le code créer, il ne reste plus qu'à l'exécuter afin de créer la base de donnée.

NB : Le script des clés étrangères (FK) devra être exécuté une fois les données importées.



```
-- SQL Base de données Fenêtres Aide
-- Retour arrière (rollback)
-- N/A < N/A > < N/A > < N/A >

-- none> script_P3.sql x

CREATE SEQUENCE public.cadastre_cadastre_id_seq;

-- CREATE TABLE public.cadastre (
cadastre_id INTEGER NOT NULL DEFAULT nextval('public.cadastre_cadastre_id_seq'),
code_type_voie SMALLINT NOT NULL,
code_voie VARCHAR(20) NOT NULL,
code_id_commune INTEGER NOT NULL,
prefixe_section INTEGER,
section VARCHAR(30),
numero_plan SMALLINT NOT NULL,
nature_culture VARCHAR(20),
nature_culture_speciale VARCHAR(20),
surface_terrain INTEGER,
CONSTRAINT cadastre_pk PRIMARY KEY (cadastre_id)
);

ALTER SEQUENCE public.cadastre_cadastre_id_seq OWNED BY public.cadastre.cadastre_id;

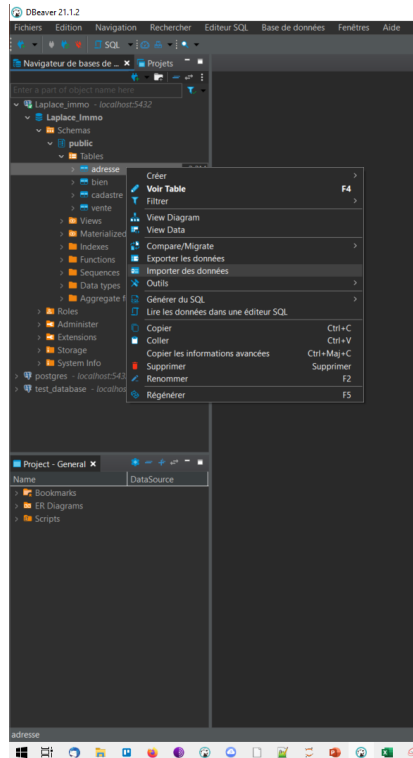
CREATE SEQUENCE public.adresse_adresse_id_seq;

-- CREATE TABLE public.adresse (
adresse_id INTEGER NOT NULL DEFAULT nextval('public.adresse_adresse_id_seq'),
type_voie VARCHAR(30),
nom_voie VARCHAR(30) NOT NULL,
code_postal INTEGER NOT NULL,
nom_commune VARCHAR(50) NOT NULL,
code_departement VARCHAR(10) NOT NULL,
CONSTRAINT adresse_pk PRIMARY KEY (adresse_id)
);
```

Chargement de la base de données

“

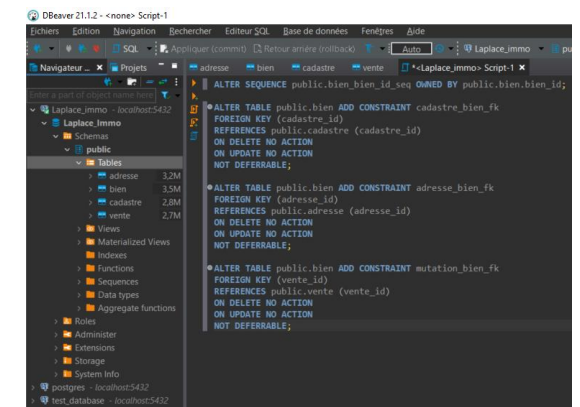
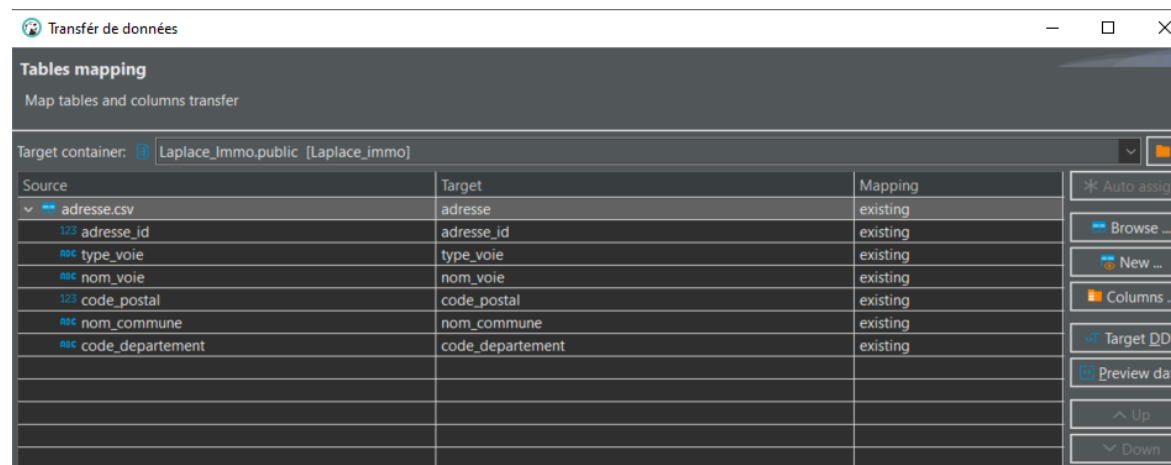
L'import des données dans la base



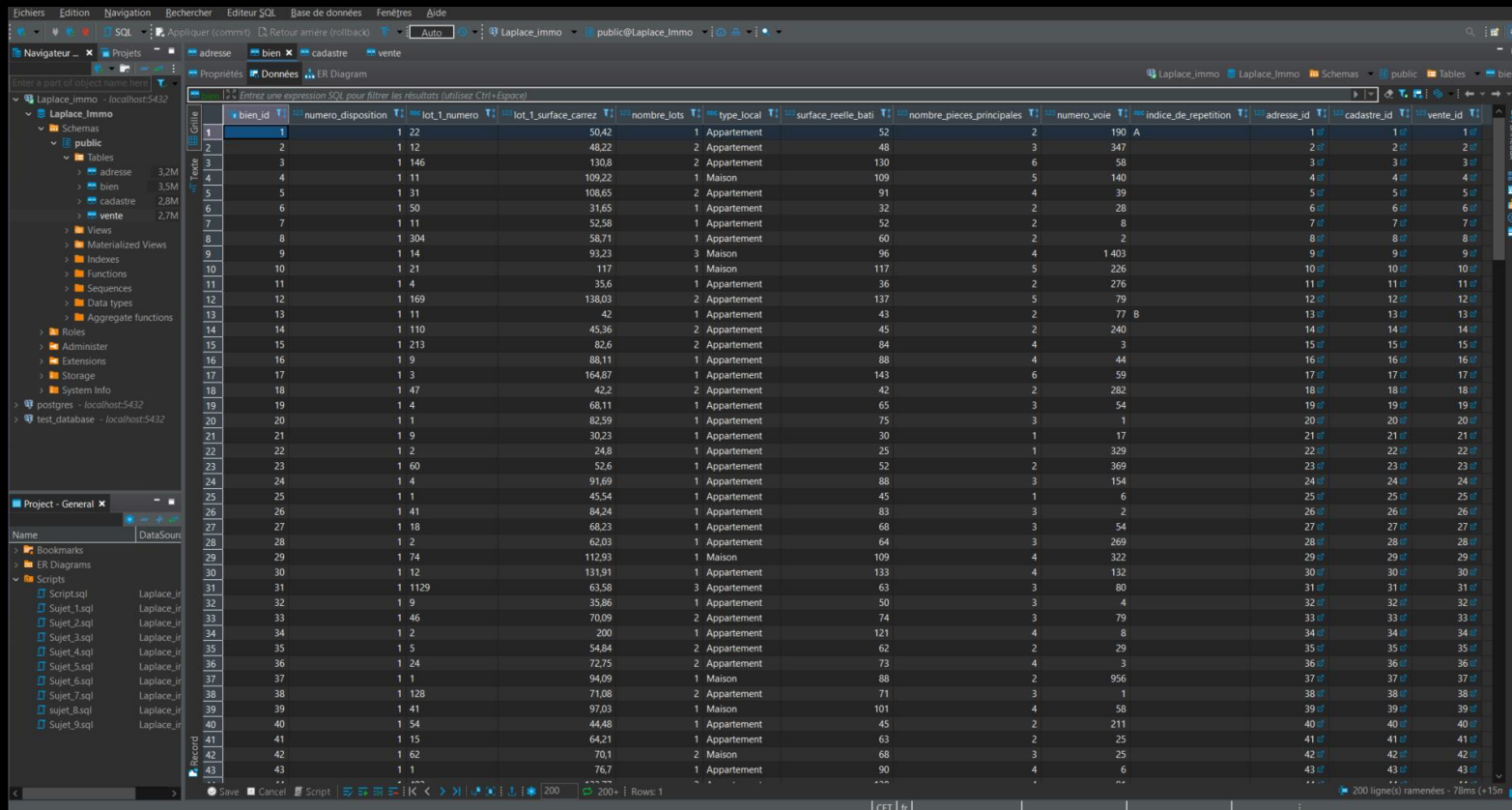
Importation des données dans le Système de Gestion de Base de Données (SGBD)

Les données sont importées dans le SGBD via les fichiers CSV de chaque table. Chaque fichier devra faire correspondre ses champs avec les tables de destination du SGBD.

NB : Le script des clés étrangères (FK) peuvent désormais être exécutés afin de créer les liens entre chaque table.



Base de données opérationnelle



The screenshot shows a PostgreSQL database interface with the 'bien' table selected. The table contains 34,169 rows of data. The columns are: bien_id, numero_disposition, lot_numero, lot_surface_carrez, nombre_lots, type_local, surface_reelle_bati, nombre_pieces_principales, numero_voie, indice_de_repetition, adresse_id, cadastre_id, and vente_id. The data is displayed in a grid format with a dark theme.

bien_id	numero_disposition	lot_numero	lot_surface_carrez	nombre_lots	type_local	surface_reelle_bati	nombre_pieces_principales	numero_voie	indice_de_repetition	adresse_id	cadastre_id	vente_id
1	22	50.42	1	Appartement	52	190	2	190	A	1	1	1
2	12	48.22	2	Appartement	48	347	3	347		2	2	2
3	146	130.8	2	Appartement	130	58	6	58		3	3	3
4	11	109.22	1	Maison	109	140	5	140		4	4	4
5	31	108.65	2	Appartement	91	39	4	39		5	5	5
6	50	31.65	1	Appartement	32	28	6	28		6	6	6
7	11	52.58	1	Appartement	52	8	7	8		7	7	7
8	304	58.71	1	Appartement	60	2	8	2		8	8	8
9	14	93.23	3	Maison	96	1403	4	1403		9	9	9
10	21	117	1	Maison	117	236	5	236		10	10	10
11	4	35.6	1	Appartement	36	276	2	276		11	11	11
12	169	138.03	2	Appartement	137	79	5	79		12	12	12
13	11	42	1	Appartement	43	77	2	77	B	13	13	13
14	110	45.36	2	Appartement	45	240	2	240		14	14	14
15	213	82.6	2	Appartement	84	3	4	3		15	15	15
16	9	88.11	1	Appartement	88	44	4	44		16	16	16
17	3	164.87	1	Appartement	143	59	6	59		17	17	17
18	47	42.2	2	Appartement	42	282	2	282		18	18	18
19	4	68.11	1	Appartement	65	54	3	54		19	19	19
20	1	82.59	1	Appartement	75	1	3	1		20	20	20
21	9	30.23	1	Appartement	30	17	1	17		21	21	21
22	2	24.8	1	Appartement	25	329	1	329		22	22	22
23	60	52.6	1	Appartement	52	369	2	369		23	23	23
24	4	91.69	1	Appartement	88	154	3	154		24	24	24
25	1	45.54	1	Appartement	45	6	1	6		25	25	25
26	41	84.24	1	Appartement	83	2	3	2		26	26	26
27	18	68.23	1	Appartement	68	54	3	54		27	27	27
28	2	62.03	1	Appartement	64	269	3	269		28	28	28
29	74	112.93	1	Maison	109	322	4	322		29	29	29
30	12	131.91	1	Appartement	133	132	4	132		30	30	30
31	1129	63.58	3	Appartement	63	80	3	80		31	31	31
32	9	35.86	1	Appartement	50	4	3	4		32	32	32
33	46	70.09	2	Appartement	74	79	3	79		33	33	33
34	2	200	1	Appartement	121	8	4	8		34	34	34
35	5	54.84	2	Appartement	62	29	2	29		35	35	35
36	24	72.75	2	Appartement	73	3	4	3		36	36	36
37	1	94.09	1	Maison	88	956	2	956		37	37	37
38	128	71.08	2	Appartement	71	1	3	1		38	38	38
39	41	97.03	1	Maison	101	58	4	58		39	39	39
40	54	44.48	1	Appartement	45	211	2	211		40	40	40
41	15	64.21	1	Appartement	63	25	2	25		41	41	41
42	62	70.1	2	Maison	68	25	3	25		42	42	42
43	1	76.7	1	Appartement	90	6	4	6		43	43	43

Base de données opérationnelle

La base de donnée est désormais opérationnelle.

Toutes les données sont importées dans leurs tables respectives, sont reliées entre elles selon le schéma relationnel et peuvent désormais faire l'objet de requêtes.

La base de données contient :

- 34169 instances

La table **ADRESSE** contient :

- 5 attributs dont 1 clé primaire

La table **BIEN** contient :

- 13 attributs dont 1 clé primaire (PK) et 3 clés étrangère (FK)

La table **CADASTRE** contient :

- 10 attributs dont 1 clé primaire (PK)

La table **VENTE** contient :

- 4 attributs dont 1 clé primaire (PK)

Requêtes

Nombre total d'appartements vendus au 1^{er} semestre 2020

```
select
from

join
where
and
;

COUNT(bien.type_local) as "Nombre d'appartements S1 2020"
bien
vente on vente.vente_id = bien.vente_id
bien.type_local = 'Appartement'
date_mutation between '2020-01-01' and '2020-06-30'
```

Requêtes 1



Contexte

Cette requête va chercher le nombre d'occurrences de l'attribut « bien.type_local » dans la table « bien ».

Fonctions utilisées

SELECT - FROM
COUNT
JOIN ON
WHERE

Jointures

Une jointure entre la table **BIEN** et la table **VENTE** est créé pour ajouter des conditions venant de la table **VENTE**.

Conditions

« bien.type_local » = 'Appartement'
« date_mutation » between '2020-01-01' and '2020-06-30'

Nombre d'appartements S1 2020
31378

Proportion des ventes d'appartements par le nombre de pièces

```
select      bien.nombre_pieces_principales as "Nombre de pieces",
            count(bien.nombre_pieces_principales)
              as "Nombre d'appartements",

            round(count(bien.nombre_pieces_principales) *100) /
              (select count(bien.nombre_pieces_principales)
               from      bien
               where      bien.type_local = 'Appartement')
              as "Proportion des ventes"

from        bien

where       bien.type_local = 'Appartement'

group by    "Nombre de pieces"
order by    "Nombre de pieces"
;
```

Requêtes 2



Contexte

Cette requête va chercher le ratio de ventes d'appartements en fonction de son nombre de pièces.

Fonctions utilisées

SELECT - FROM

COUNT

ROUND(COUNT (opération avec sous requête))

WHERE

GROUP BY - ORDER BY

Conditions

« bien.type_local » = 'Appartement'

Groupeement

« Nombre de pieces »

Ordre

« Nombre de pieces » - Ascendant

Nombre de pieces	Nombre d'appartements	Proportion des ventes
0	30	0.0956083880425776
1	6739	21.476830900631015
2	9783	31.177895340684557
3	8966	28.574160239658358
4	4460	14.213780355663204
5	1114	3.550258142647715
6	204	0.6501370386895277
7	54	0.17209509847663967
8	17	0.05417808655746064
9	8	0.02549557014468736
10	2	0.00637389253617184
11	1	0.00318694626808592

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

```
select      adresse.code_departement as "Departement",
            round(avg (vente.valeur_fonciere /
            bien.lot_1_surface_carrez),2) as "Prix moyen du m2"

from        vente

join        bien on vente.vente_id = bien.vente_id
join        adresse on adresse.adresse_id=bien.adresse_id

group by    "Departement"
order by    "Prix moyen du m2" desc
limit       10
;
```

Requêtes 3



Contexte

Cette requête va chercher le prix moyen du mètre carré de chaque département et limiter le résultat sur 10 d'entre eux.

Fonctions utilisées

SELECT - FROM
ROUND(AVG (opération))
COUNT
JOIN ON
GROUP BY - ORDER BY
LIMIT

Jointures

Une jointure entre les tables **VENTE**, **BIEN** et **ADRESSE** est créée pour ajouter des conditions venant de la table **ADRESSE**.

Groupe ment

« Département »

Ordre de tri

« Prix moyen du m2 » - Descendant

Departement	Prix moyen du m2
75	12052,89
92	7219,39
94	5343,28
6	4700,33
74	4667,13
93	4344,78
78	4225,25
69	4059,31
2A	4026,97
33	3764,14

Prix moyen du mètre carré d'une maison en Ile de France

```
select      (round(avg(vente.valeur_fonciere/bien.lot_1_surface_carrez),2))
            as "Prix moyen m2 en Ile de France"
from        adresse

join        bien on bien.vente_id =adresse.adresse_id
join        vente on vente.vente_id =bien.adresse_id

where       adresse.code_departement in ('75','77','78','91','92','93','94','95')
and         bien.type_local = 'Maison'

;
```

Requêtes 4



Contexte

Cette requête va chercher le prix moyen du mètre carré pour les maisons situées dans les départements de l'Ile de France.

Fonctions utilisées

SELECT - FROM
ROUND(AVG (opération))
JOIN ON
WHERE - AND

Jointures

Une jointure entre les tables **ADRESSE**, **BIEN** et **VENTE** est créée pour ajouter des conditions venant de la table **VENTE**.

Conditions

«adresse.code_departement» in ('75','77','78','91','92','93','94','95')
« bien.type_local » = 'Maison'

Prix moyen du m2 en Ile de France
3745.01

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

```
select      bien.bien_id as "ID appartements",
            adresse.code_departement as "Departement",
            vente.valeur_fonciere as "Valeur fonciere",
            bien.lot_1_surface_carrez as "Surface Carrez"

from        bien

join        vente on bien.vente_id = vente.vente_id
join        adresse on bien.adresse_id = adresse.adresse_id

where       bien.type_local = 'Appartement'
and         vente.valeur_fonciere is not null

group by    "ID appartements",
            "Departement",
            "Valeur fonciere",
            "Surface Carrez"

order by    "Valeur fonciere" desc
limit      10
;
```

ID appartements	Departement	Valeur fonciere	Surface Carrez
32275	75	9000000.00	9.10
21835	91	8600000.00	64.00
29799	75	8577713.00	20.55
32433	75	7620000.00	42.77
29850	75	7600000.00	253.30
29522	75	7535000.00	139.90
31973	75	7420000.00	360.95
32135	75	7200000.00	595.00
29353	75	7050000.00	122.56
29513	75	6600000.00	79.38

Requêtes 5



Contexte

Cette requête va chercher la liste des 10 appartements les plus chers et afficher l'ID, le Département, la Valeur Foncière et la Surface Carrez.

Fonctions utilisées

SELECT - FROM
JOIN
WHERE - AND
GROUP BY - ORDER BY
LIMIT

Jointures

Une jointure entre les tables **BIEN**, **VENTE** et **ADRESSE** est créée pour ajouter des conditions venant de la table **VENTE**.

Conditions

« bien.type_local » = 'Appartement'
« Valeur fonciere » is not null

Groupe

ID appartements, Département, Valeur Foncière, Surface Carrez

Ordre

« Valeur fonciere »

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

```
with
table1      as (
select      round(count(vente.vente_id),2) as nb_ventes_trimestre_1
from        vente
where       vente.date_mutation between '2020-01-01' and '2020-03-31'),

table2      as (
select      round(count(vente.vente_id),2) as nb_ventes_trimestre_2
from        vente
where       vente.date_mutation between '2020-04-01' and '2020-06-30')

select      round((((table2.nb_ventes_trimestre_2 -
table1.nb_ventes_trimestre_1)/table1.nb_ventes_trimestre_1*100),2)
              as "Taux d'evolution"
from        table1, table2
;
```

Requêtes 6



Contexte

Cette requête va chercher la ratio de ventes entre le 1er et le 2nd trimestre

Fonctions utilisées

WITH
SELECT - FROM
ROUND(COUNT (opération))
WHERE - AND

Tables temporaires

2 tables intermédiaires sont créées avec chacune leur conditions respectives afin de pouvoir faire l'opération qui conclura au ratio des ventes dans un nouveau SELECT.

Conditions

vente.date_mutation between '2020-01-01' and '2020-03-31'
vente.date_mutation between '2020-04-01' and '2020-06-30'

Taux d'evolution	
	3.68

Liste des communes où le nombre de ventes a augmenté d'au moins 20% entre le premier et le second trimestre de 2020

----- Tables temporaires

with

table1 as (

select adresse.nom_commune,
 round(count(vente.vente_id),2) as nb_ventes_1

from adresse

join bien on bien.adresse_id = adresse.adresse_id
join vente on bien.vente_id = vente.vente_id

where vente.date_mutation between '2020-01-01' and '2020-03-31'
group by adresse.nom_commune),

table2 as (

select adresse.nom_commune,
 round(count(vente.vente_id),2) as nb_ventes_2

from adresse

join bien on bien.adresse_id = adresse.adresse_id
join vente on bien.vente_id = vente.vente_id

where vente.date_mutation between '2020-04-01' and '2020-06-30'
group by adresse.nom_commune)

----- Requête select

select table1.nom_commune as "Communes",
 table1.nb_ventes_1 as "Nombre de ventes trimestre 1",
 table2.nb_ventes_2 as "Nombre de ventes trimestre 2",
 ROUND((table2.nb_ventes_2 - table1.nb_ventes_1)/table1.nb_ventes_1*100,2) as "Evolution"

from table1
join table2 on table1.nom_commune = table2.nom_commune

where round((((table2.nb_ventes_2 - table1.nb_ventes_1)/table1.nb_ventes_1)*100) >= 20
order by "Evolution"
;

Requêtes 7



Contexte

Cette requête va chercher la différence en pourcentage des ventes sur chaque commune entre le 1^{er} et le 2nd trimestre 2020. Les résultats seront limités aux communes excédant 20%.

Fonctions utilisées

WITH
SELECT - FROM
ROUND(COUNT (opération))
JOIN ON
WHERE
GROUP BY
ORDER BY

Tables temporaires

2 tables intermédiaires sont créées avec chacune leur conditions respectives afin de pouvoir effectuer l'opération qui calculera la différence des ventes dans un nouveau SELECT.

Jointures

Une jointure entre les tables **ADRESSE**, **BIEN** et **VENTE** est créée pour chaque table temporaire pour ajouter des conditions venant de la table **VENTE**.

Conditions

« date_mutation » between '2020-01-01' and '2020-03-31'
« date_mutation » between '2020-04-01' and '2020-06-30'

Groupeement

« adresse.nom_commune »

Ordre

« Evolution » - Ascendant

Communes	Nombre de ventes trimestre 1	Nombre de ventes trimestre 2	Evolution
BESSANCOURT	5.00	6.00	20.00
LE BEAUSSET	10.00	12.00	20.00
SCHOELCHER	5.00	6.00	20.00
SAINT-HILAIRE-DE-RIEZ	15.00	18.00	20.00
CHESSY	5.00	6.00	20.00
GOLBEY	5.00	6.00	20.00
LA QUEUE-EN-BRIE	10.00	12.00	20.00
SAINTE FOY LES LYON	5.00	6.00	20.00
DIVONNE-LES-BAINS	5.00	6.00	20.00
GIVORS	5.00	6.00	20.00
BRIANCON	5.00	6.00	20.00
ARCACHON	25.00	30.00	20.00
LEUCATE	29.00	35.00	20.69
ETAMPES	24.00	29.00	20.83
LE CHESNAY-ROCQUENCOURT	14.00	17.00	21.43
BOISSY-SAINT-LEGER	9.00	11.00	22.22
ORLY	9.00	11.00	22.22
PARIS 08	62.00	77.00	24.19
RENNES	61.00	76.00	24.59
L'ILE-SAINT-DENIS	4.00	5.00	25.00
MERY-SUR-OISE	4.00	5.00	25.00
BONNEVILLE	4.00	5.00	25.00
VAL DE BRIEY	4.00	5.00	25.00
VALENCE	28.00	35.00	25.00
LONGPONT-SUR-ORGE	4.00	5.00	25.00
PONTOISE	24.00	30.00	25.00
BATZ-SUR-MER	4.00	5.00	25.00
SOUSTONS	8.00	10.00	25.00
CHAMONIX MONT BLANC	12.00	15.00	25.00
SAINT-GENIS-LAVAL	8.00	10.00	25.00
NOGENT SUR OISE	8.00	10.00	25.00
CADAUJAC	4.00	5.00	25.00
PARIS 11	169.00	214.00	26.63
VILLEMOMBLE	15.00	19.00	26.67
SARCELLES	15.00	19.00	26.67
NICE	173.00	220.00	27.17
LAGNY	11.00	14.00	27.27

Résultat requêtes 7



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

```
with
table1      as (
select      round(avg(vente.valeur_fonciere/bien.lot_1_surface_carrez),2)
              as valfonc_moy_2

from        adresse

join        bien on bien.vente_id =adresse.adresse_id
join        vente on vente.vente_id =bien.adresse_id

where       bien.nombre_pieces_principales = 2
and         bien.type_local ='Appartement'),

table2      as (
select      round(avg(vente.valeur_fonciere/bien.lot_1_surface_carrez),2)
              as valfonc_moy_3

from        adresse

join        bien on bien.vente_id =adresse.adresse_id
join        vente on vente.vente_id =bien.adresse_id

where       bien.nombre_pieces_principales = 3
and         bien.type_local ='Appartement')

select      round((((table2.valfonc_moy_3 - table1.valfonc_moy_2)/table1.valfonc_moy_2)*100,2)
              as "Taux d'evolution"

from        table1, table2
;
```

Requêtes 8



Contexte

Cette requête va chercher la ratio de ventes entre le 1er et le 2nd trimestre

Fonctions utilisées

WITH
SELECT - FROM
ROUND(AVG (opération))
WHERE - AND

Jointures

2 tables intermédiaires sont créées avec chacune leur conditions respectives afin de pouvoir faire l'opération qui conclura au ratio des ventes dans un nouveau SELECT.

Conditions

bien.type_local ='Appartement'
bien.nombre_pieces_principales = 3 / 2

Taux d'evolution	
	-12.40

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

```
with table1 as (  
  
  select      adresse.code_departement as "Departement",  
              adresse.nom_commune as "Commune",  
              round(avg(vente.valeur_fonciere),2) as "Valeur fonciere",  
              rank() over(partition by adresse.code_departement  
                           order by round(avg(vente.valeur_fonciere),2) desc)  
              as "Classement"  
  
  from        adresse  
  
  join        bien on bien.adresse_id= adresse.adresse_id  
  join        vente on vente.vente_id=bien.vente_id  
  
  where       adresse.code_departement in ('6','13','33','59','69')  
  and         vente.valeur_fonciere is not null  
  
  group by    "Departement", "Commune"  
  order by    "Valeur fonciere" desc)  
  
select *  
from table1  
where "Classement" <= 3  
order by "Departement","Valeur fonciere" desc  
;
```

Requêtes 9



Contexte

Cette requête va chercher la moyenne des valeurs foncières, pour les 3 communes les plus chers des départements 6, 13, 33, 59 et 69.

Fonctions utilisées

WITH
SELECT - FROM
ROUND(AVG (opération))
RANK() OVER(PARTITION BY)
JOIN ON
WHERE
GROUP BY
ORDER BY

Tables temporaires

1 table temporaire est créée pour créer un classement des communes avec les valeurs foncières les plus élevées.

Jointures

Une jointure entre les tables **ADRESSE**, **BIEN** et **VENTE** est créé pour chaque table temporaire pour ajouter des conditions venant de la table **VENTE**.

Conditions

« adresse.code_departement » in ('6','13','33','59','69')
"Classement" <= 3

Groupeement

"Departement", "Commune"

Ordre

"Departement","Valeur fonciere" – Descendant

Résultat requêtes 9

Departement	Commune	Valeur fonciere	Classement
13	GIGNAC-LA-NERTHE	330000.00	1
13	SAINT SAVOURNIN	314425.00	2
13	CASSIS	313416.88	3
33	LEGE-CAP-FERRET	549500.64	1
33	VAYRES	335000.00	2
33	ARCACHON	307435.93	3
59	BERSEE	433202.00	1
59	CYSOING	408550.00	2
59	HALLUIN	322250.00	3
6	SAINT-JEAN-CAP-FERRAT	968750.00	1
6	EZE	655000.00	2
6	MOUANS-SARTOUX	476898.10	3
69	VILLE SUR JARNIOUX	485300.00	1
69	LYON 2EME	455217.27	2
69	LYON 6EME	426968.25	3



Merci
de votre attention

