OCR - PROJET 3

Création et utilisation d'une base de données permettant de suivre les transactions immobilières en France

Mathilde LE SOLLIEC

Juillet 2025

SOMMAIRE

1. Contexte

Analyse des transactions immobilières

Un réseau national d'agence immobilières, Laplace immo, souhaite collecter l'ensemble des transactions immobilières en France afin de :

- suivre l'évolution du prix au mètre carré,
- d'identifier les régions où le marché est le plus porteurs.

- Projet : en tant que Data Engineer
- Créer une base de données qui permet de répondre aux interrogations métiers sur les transactions immobilière
- Réalisation d'une Proof of Concept (POC) avec les années 2020, avant la mise en oeuvre complète

2. Les données

Sources de données

3 tables

Valeurs-foncières

- Informations sur les transactions et sur les biens (prix, caractéristiques du bien, date de la vente...)
- 1 ligne = 1 transaction
- 46 colonnes x 34 179 lignes

donnees_communes

Données INSEE
Chaque ligne par communes
Info sur les département, les populations...

- 9 colonnes X 34 992 lignes

fr-esr-referentiel-geographique

Données géo-référentiel plus complète sur les communes

- 21 colonnes X 38 917

Tri des données

Stratégie de sauvegarde et conformité RGPD

1. Objectif défini : faire des statistiques fiables pour suivre l'évolution du marché immobilier
2. Garder les données utiles pour répondre à notre besoin métier
3. Données anonymisées
4. Ne sera pas partagé

→ pourrait faire de la pseudonymisation (ex : remplacer la date exacte par le mois, arrondir les valeurs identifiable)

Création d'un dictionnaire de données

Extraits du dictionnaire de données

CODE	DEFINITION	TYPE	LONGUEUR	NATURE	REGLE DE GESTION	REGLE DE CALCUL
sale_id	Identifiant de la transaction immobilièr	Integer	NC	Elémentaire	Ne doit pas être nul, doit être ur	**généré / autoincrémenté**
date	Date de la transaction - de la vente	Date	NC	Elémentaire	Format de la date (aaaa-mm-jj)	
property_value	Valeur foncière du bien en euros	Float	NC	Elémentaire		
property_id	Identifiant de la propriété	Integer	NC	Elémentaire		

2. Le schéma relationnel normalisé

Méthodologie

1. Liste des données à modéliser
2. Chaque valeur est atomique
   1 colonne = 1 informations (pas de liste)
   “Une cellule contient-elle plusieurs infos ?”
3. Une clé primaire pour chaque table
   Clé primaire unique
4. Chaque attribut dépends de la clé primaire
   Séparation des tables - si dépendances partielles
5. Chaque attribut dépends **uniquement** de la clé primaire
   Séparation des tables - si dépendances transitives
   “Cette colonne dépend-elle seulement de la clé ?”
6. Ajout des relations entre les tables
   Utilisation des clés étrangères
7. Schéma relationnel normalisé

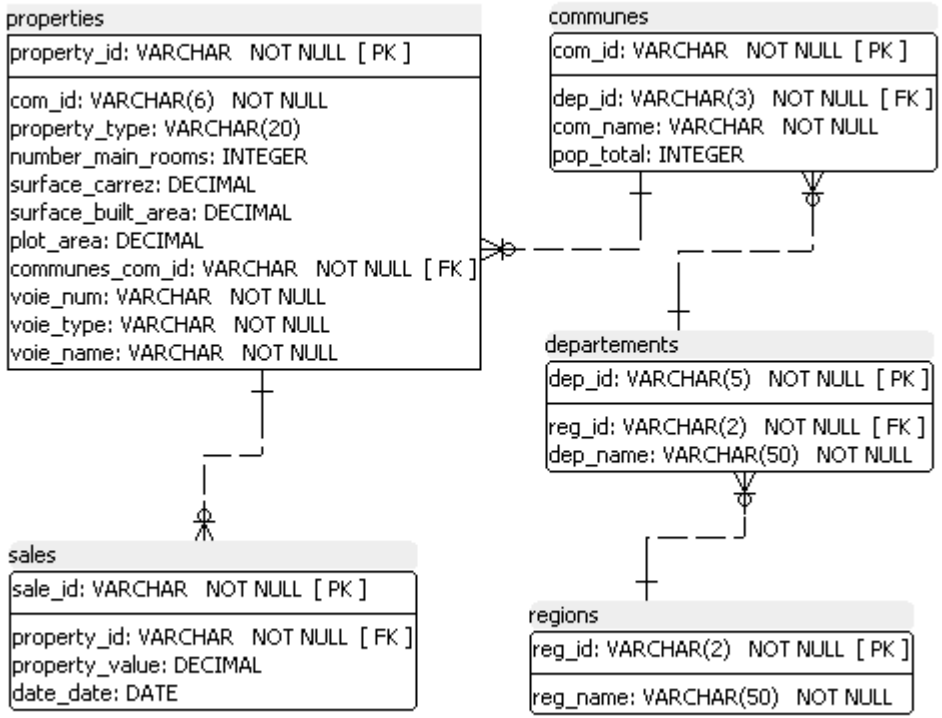
Résultat

5 tables

Properties : information sur les biens
Sales : information sur les ventes
Communes
Départements
Région

Schéma relationnel 3NF

Pas de redondance, d'incohérence
Facilité de maintenance et d'évolution



3. mySQL

Préparation des tables

1

Préparation des table CSV :
- nom des tables
- colonnes
- types de données

MySQL

2

Création des tables dans mySQL

```
CREATE TABLE sales(
sale_id VARCHAR(50) NOT NULL,
property_id VARCHAR(50) NOT NULL,
date_date DATE NOT NULL,
property_value DECIMAL(10,2),
PRIMARY KEY (sale_id)
);

ALTER TABLE sales ADD CONSTRAINT properties_sales_fk
FOREIGN KEY (property_id)
REFERENCES properties (property_id)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

3

Téléchargement des tables dans mySQL

```
LOAD DATA LOCAL INFILE 'C:/Users/mathi/OneDrive/Documents/
INTO TABLE sales
FIELDS TERMINATED BY ';'
IGNORE 1 LINES
(sale_id, property_id, date_date, property_value)
SET property_value = NULLIF(property_value, '');
```

Les requêtes
&
résultats des
interrogations métiers

1

Question et résultat

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

31 378

Requête

```
SELECT COUNT(sale_id) as nbr_appartement_sale
FROM projet3_ocr_immo.sales as s
LEFT JOIN projet3_ocr_immo.properties as p
USING(property_id)
WHERE p.property_type LIKE "Appartement"
    AND MONTH(date_date) BETWEEN 1 AND 6
    AND YEAR(date_date) = 2020;
```

2

Question et résultat

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

nbr_appartement_sale	reg_name
13995	Ile-de-France
3649	Provence-Alpes-Côte d'Azur
3253	Auvergne-Rhône-Alpes
1932	Nouvelle-Aquitaine
1640	Occitanie
1357	Pays de la Loire
1254	Hauts-de-France
984	Grand Est
983	Bretagne
862	Normandie
696	Centre-Val de Loire
376	Bourgogne-Franche-Comté
223	Corse
94	Martinique
44	La Réunion
34	Guyane
2	Guadeloupe

Requête

```sql
WITH sales_info2 AS (
    SELECT
        s.sale_id,
        s.date_date,
        p.property_type,
        r.reg_name
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id
        )

SELECT
    COUNT(sale_id) as nbr_appartement_sale
    , reg_name
FROM sales_info2
WHERE property_type LIKE "Appartement"
    AND MONTH(date_date) BETWEEN 1 AND 6
    AND YEAR(date_date) = 2020
GROUP BY reg_name
ORDER BY COUNT(sale_id) DESC;
```

3

Question et résultat

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

number_main_rooms	nbr_appartement_sale	proportion_percent
2	9783	31.18
3	8966	28.57
1	6739	21.48
4	4460	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

Requête

```sql
WITH sales_info3 AS (
    SELECT
        s.sale_id,
        p.property_type,
        p.number_main_rooms
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id )

SELECT
    number_main_rooms,
    COUNT(sale_id) AS nbr_appartement_sale,
    -- Windows function sur le total de sale pour l'utiliser sur le calcul de la proportion
    ROUND(COUNT(sale_id) / SUM(COUNT(sale_id)) OVER () * 100.0 , 2) AS proportion_percent
FROM sales_info3
WHERE property_type LIKE "Appartement"
GROUP BY number_main_rooms
ORDER BY proportion_percent DESC;
```

4

Question et résultat

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

dep_name	avg_price_m2
Paris	12053
Hauts-de-Seine	7219
Val-de-Marne	5343
Alpes-Maritimes	4700
Haute-Savoie	4667
Seine-Saint-Denis	4345
Yvelines	4225
Rhône	4059
Corse-du-Sud	4027
Gironde	3764

Requête

```
WITH sales_info4 AS(
    SELECT
        s.sale_id,
        s.property_value,
        p.surface_carrez,
        d.dep_name
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id )

SELECT
  dep_name,
 ROUND(AVG(property_value / surface_carrez),0) as avg_price_m2
FROM sales_info4
GROUP BY dep_name
ORDER BY AVG(property_value / surface_carrez)  DESC
LIMIT 10;
```

5

Question et résultat

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

3 745

Requête

```sql
WITH sales_info5 AS(
    SELECT
        s.sale_id,
        s.property_value,
        p.property_type,
        p.surface_carrez,
        r.reg_name
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id)

SELECT
    ROUND(AVG(property_value / surface_carrez),0) as avg_price_m2_IDF_House
FROM sales_info5
WHERE reg_name LIKE "Ile-de-France"
    AND property_type LIKE "Maison" ;
```

6

Question et résultat

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

property_id	property_value	reg_name	surface_carrez
P30603	9000000.00	Ile-de-France	9.10
P5261	8600000.00	Ile-de-France	64.00
P3625	8577713.00	Ile-de-France	20.55
P7602	7620000.00	Ile-de-France	42.77
P9988	7600000.00	Ile-de-France	253.30
P17823	7535000.00	Ile-de-France	139.90
P410	7420000.00	Ile-de-France	360.95
P16357	7200000.00	Ile-de-France	595.00
P1924	7050000.00	Ile-de-France	122.56
P19161	6600000.00	Ile-de-France	79.38

Requête

```sql
WITH sales_info6 AS (
    SELECT
        s.sale_id,
        s.property_id,
        s.property_value,
        p.property_type,
        p.surface_carrez,
        r.reg_name
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id )

SELECT
    property_id,
    property_value,
    reg_name,
    surface_carrez
FROM sales_info6
WHERE property_type LIKE "Appartement"
ORDER BY property_value DESC
LIMIT 10;
```

7

Question et résultat

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

+3,55%

```sql
-- 7 - Taux d'évolution du nombre de ventes entre le premier et le second trimestre de 2020
WITH sales_info7 AS (
    SELECT
        s.sale_id,
        s.date_date,
        p.property_type,
        r.reg_name
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id),

 first_trimestre_sales AS (
    SELECT COUNT(sale_id) as nbr_sale_first_trimestre
    FROM sales_info7
    WHERE MONTH(date_date) BETWEEN 1 AND 3
        AND YEAR(date_date) = 2020),

second_trimestre_sales AS (
    SELECT COUNT(sale_id) as nbr_sale_second_trimestre
    FROM sales_info7
    WHERE MONTH(date_date) BETWEEN 4 AND 6
        AND YEAR(date_date) = 2020
        )

SELECT
    round (
        (nbr_sale_second_trimestre - nbr_sale_first_trimestre) / nbr_sale_second_trimestre * 100 , 2
        )AS evol_sales_first_second_trimestre
    FROM first_trimestre_sales
        JOIN second_trimestre_sales
```

8

Question et résultat

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

reg_name	avg_price_m2
Ile-de-France	8770
La Réunion	3642
Provence-Alpes-Côte d'Azur	3588
Corse	3105
Auvergne-Rhône-Alpes	2891
Nouvelle-Aquitaine	2465
Bretagne	2412
Pays de la Loire	2316
Hauts-de-France	2190
Occitanie	2097
Normandie	2016
Grand Est	1541
Centre-Val de Loire	1453
Bourgogne-Franche-Comté	1251
Martinique	573

Requête

```sql
WITH sales_info8 AS (
    SELECT
        s.sale_id,
        s.property_value,
        p.property_type,
        p.surface_carrez,
        r.reg_name,
        p.number_main_rooms
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id )

SELECT
    reg_name,
    ROUND(AVG(property_value / surface_carrez),0) as avg_price_m2
FROM sales_info8
WHERE number_main_rooms > 4
    AND property_type LIKE "Appartement"
GROUP BY reg_name
ORDER BY ROUND(AVG(property_value / surface_carrez),0) DESC;
```

9

Question et résultat

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

com_name	nbr_sale_first_semestre
Montreuil	138
Rennes	137
Paris 7e Arrondissement	135
Paris 13e Arrondissement	131
Angers	127
Paris 2e Arrondissement	127
Paris 5e Arrondissement	124
Saint-Etienne	123
Neuilly-sur-Seine	123
Paris 4e Arrondissement	120
Saint-Maur-des-Fossés	116
Versailles	114
Hyères	114
Le Havre	113
Saint-Denis	113
Rueil-Malmaison	111
Rouen	110
Clichy	109
Ajaccio	104
Nancy	102
Maisons-Alfort	101

Requête

```sql
WITH sales_info9 AS(
    SELECT
        s.sale_id,
        c.com_name,
        s.date_date
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id)

SELECT
    com_name,
    COUNT(sale_id) as nbr_sale_first_semestre
FROM sales_info9
WHERE MONTH(date_date) BETWEEN 1 AND 6
    AND YEAR(date_date) = 2020
GROUP BY com_name
HAVING COUNT(sale_id) > 50
ORDER BY COUNT(sale_id) DESC ;
```

10

Question et résultat

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

5,95%

Requête

```sql
WITH sales_info10 AS (
    SELECT
        p.number_main_rooms,
        s.sale_id,
        s.property_value,
        p.property_type,
        p.surface_carrez
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id )

, pricem2_room2 AS (
    SELECT ROUND(AVG(property_value / surface_carrez),0) as avg_price_m2_rooms2
    FROM sales_info10
    WHERE number_main_rooms = 2
        AND property_type LIKE "Appartement"
        )

, pricem2_room3 AS (
    SELECT ROUND(AVG(property_value / surface_carrez),0) as avg_price_m2_rooms3
    FROM sales_info10
    WHERE number_main_rooms IN (3,2)
        AND property_type LIKE "Appartement"
        )

SELECT
  ROUND(
    ((avg_price_m2_rooms2 - avg_price_m2_rooms3) / avg_price_m2_rooms2) * 100,
    2
  ) AS diff_price_room2_3_percent
FROM pricem2_room2 CROSS JOIN pricem2_room3;
-- pas sure du cross join : ne fonctionne que parce qu'il y en a deux
```

11

Question et résultat

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

dep_id	com_name	rank_dep	avg_property_value
13	Gignac-la-Nerthe	1	330000
13	Saint-Savournin	2	314425
13	Cassis	3	313417
33	Lège-Cap-Ferret	1	549501
33	Vayres	2	335000
33	Arcachon	3	307436
59	Bersée	1	433202
59	Cysoing	2	408550
59	Halluin	3	322250
6	Saint-Jean-Cap-Ferrat	1	968750
6	Eze	2	655000
6	Mouans-Sartoux	3	476898
69	Ville-sur-Jarnioux	1	485300
69	Lyon 2e Arrondissement	2	455217
69	Lyon 6e Arrondissement	3	426968

Requête

```sql
WITH sales_info11 AS (
    SELECT
        s.sale_id,
        s.property_value,
        c.com_name,
        d.dep_id
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id
        )

, avg_value_com AS (
    SELECT
        ROUND(AVG(property_value),0) as avg_property_value,
        com_name,
        dep_id
    FROM sales_info11
        WHERE dep_id IN (6, 13, 33, 59, 69)
        GROUP BY com_name, dep_id
        )

, rank_com AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY dep_id ORDER BY avg_property_value DESC) AS rank_dep
    FROM avg_value_com
    )

SELECT
    dep_id,
    com_name,
    rank_dep,
    avg_property_value
FROM rank_com
WHERE rank_dep <= 3
ORDER BY dep_id, avg_property_value DESC;
```

12

Question et résultat

Les 20 communes avec le plus de transactions pour 1000 habitants

com_name	sales_per_1000
Germ	55.56
Chamrousse	36.41
Isola	33.08
Villers-sur-Mer	28.45
Eaux-Bonnes	25.51
Gresse-en-Vercors	20.57
Allos	20.43
Enchastrayes	20.00
Tourgéville	19.98
Houlgate	19.45
Uvernet-Fours	18.09
Huez	16.42
Puyvalador	15.63
Péone	15.15
Gruissan	14.52

Requête

```
WITH sales_info12 AS (
    SELECT
        s.sale_id,
        c.com_name,
        d.dep_id,
        c.pop_total
    FROM projet3_ocr_immo.sales AS s
    JOIN projet3_ocr_immo.properties AS p
        ON s.property_id = p.property_id
    JOIN projet3_ocr_immo.communes AS c
        ON p.com_id = c.com_id
    JOIN projet3_ocr_immo.departements AS d
        ON c.dep_id = d.dep_id
    JOIN projet3_ocr_immo.regions AS r
        ON d.reg_id = r.reg_id
    )

,transactions_per_com AS (
    SELECT
        com_name,
        COUNT(sale_id) AS nbr_sales,
        MAX(pop_total) AS population
    FROM sales_info12
    GROUP BY com_name
)
```

```
SELECT
    com_name,
    ROUND(nbr_sales / population * 1000, 2) AS sales_per_1000
FROM transactions_per_com
ORDER BY sales_per_1000 DESC
LIMIT 20;
```

Difficultés rencontrées

Difficultés

Formatage des données avant l'import ;

1. VARCHAR (N) : trop court → écarte des données pendant le téléchargement
2. Gestion des NULL → j'ai du passer par la commande
3. Date devait être au format AAAA-MM-DD

Si c'était à refaire :

1- Utilisation de Python plutôt que Excel pour le formatage → permet plus facilement de manipuler et de voir les max par exemple pour le VARCHAR, transformer les données

2- L'import des données beaucoup plus rapide en utilisant la commande SQL au lieu sur l'interface graphique (menus, boutons)