Création et utilisation de la base de données

Charlotte Yung

Laplace Immo

Contexte du projet

Anticiper le Marché Immobilier grâce aux Données

- Le projet "DATAImmo" chez Laplace Immo vise à développer un modèle de prévision des prix de vente pour se démarquer de la concurrence. Il inclut l'optimisation de la base de données des transactions immobilières en France, afin d'analyser le marché et de soutenir les agences régionales.

La stratégie de sauvegarde et la conformité RGPD

- **Stratégie de sauvegarde**

Mise en place d'une stratégie de sauvegarde régulière pour sécuriser les données, incluant des sauvegardes sur serveurs et cloud, ainsi que des tests de restauration pour garantir leur récupération en cas d'incident.

- **Conformité RGPD**

Tri des fichiers Excel et anonymisation des données sensibles, comme les 'noms des acquéreurs', afin de garantir la conformité avec les exigences du RGPD.

Les données initiales

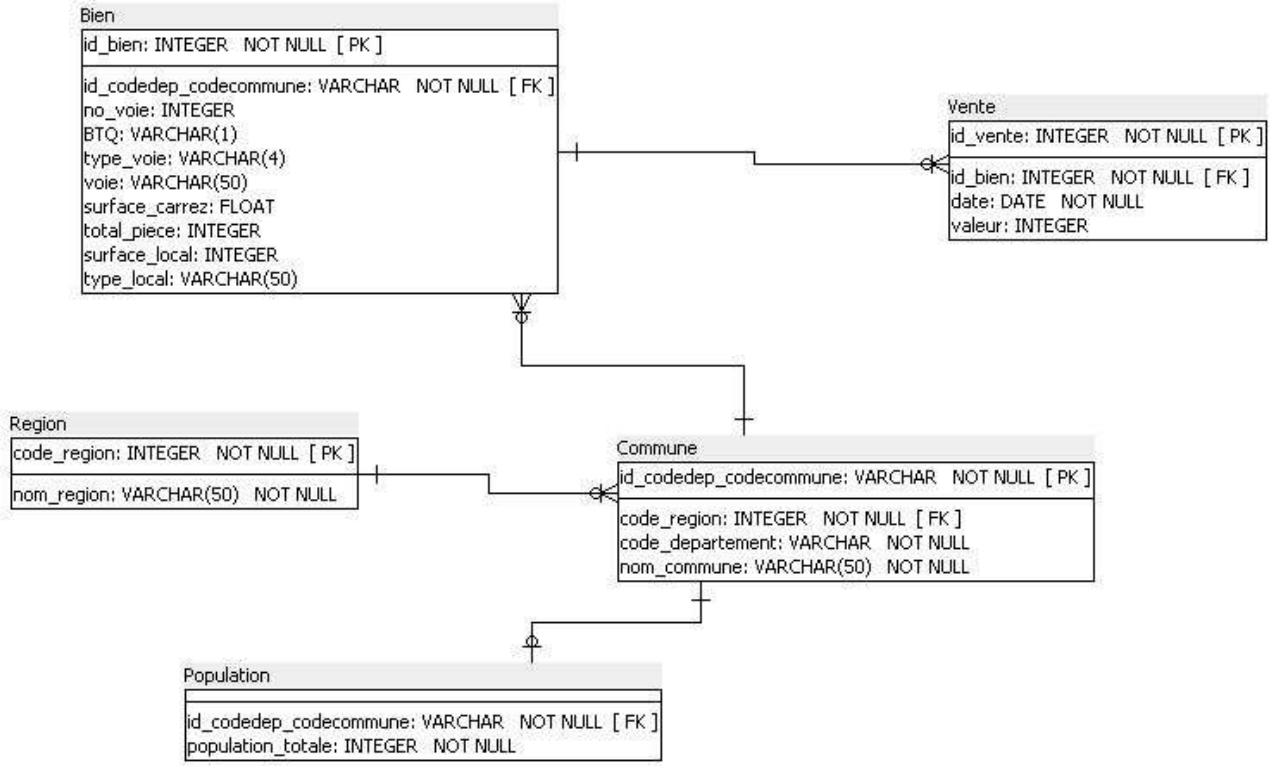
	Valeurs foncières	Référenciel Géographique	Communes
Source	Données open data des DVF	Données de data.gouv	Données INSEE
Qualité	- Vérifications valeurs manquantes - Nettoyage doublons	Ajustements nécessaires pour l'intégration	Vérification de la MAJ des données démographiques
Types	Transactions foncières (prix, surface, localisation, …)	Données géographiques des zones administratives	Données démographiques par commune
Problèmes	Conformité RGPD (anonymisation des données sensibles)	Harmonisation des formats pour le modèle de BDD	Aucun

L’extrait du dictionnaire des données

Extrait du dictionnaire des données « Valeurs Foncières »

CODE	SIGNIFICATION	TYPE	LONGUEUR	NATURE	REGLE DE GESTION	REGLE DE CALCUL
Id_bien	Identifiant unique du bien	INTEGER	NC	Clé primaire	NOT NULL	
id_codedep_codecommune	Identifiant unique des communes	VARCHAR	5	Clé étrangère	NOT NULL	Concaténation des colonnes "Code departement" et "code commune"
No_voie	Numéro de la voie	INTEGER	4	Elémentaire		
BTQ	Complément du numéro de la voie	VARCHAR	1	Elémentaire		
Type_voie	Type de voie	VARCHAR	4	Elémentaire		
Voie	Nom de la voie	VARCHAR	NC	Elémentaire		
Total_pièce	Nombre total de pièces	INTEGER	2	Elémentaire	≥ 0	
Surface_carrez	Surface du bien mesurée en loi carrez (m2)	FLOAT	NC	Elémentaire	≥ 0	
Surface_local	Surface total du local (m2)	INTEGER	NC	Elémentaire		
Type_local	Type de local	VARCHAR	NC	Elémentaire		
Id_vente	Identifiant unique de la vente	INTEGER		Clé primaire	NOT NULL	
Date	Date de signature de l'acte de vente	DATE		Elémentaire	Format : AAA/MM/JJ	
Valeur	Montant de la transaction (en euros)	INTEGER		Elémentaire	≥ 0	

Le schéma relationnel normalisé



La base de données avec les tables créées et les données chargées

- Table « Bien » :

- Table « Commune » :

- Table « Population » :

- Table « Région » :

- Table « Vente » :

Les requêtes ou screenshot qui permettent de démontrer le bon chargement des données

- Vérification des tables :

```
SELECT name FROM sqlite_master WHERE type='table';
```

	name
1	Region
2	Population
3	Commune
4	Bien
5	Vente

- Nombre d'enregistrements :

```
-- Nombre d'enregistrements dans la table "Vente"
SELECT COUNT(*) AS "Nombre de lignes" FROM Vente;

-- Nombre d'enregistrements dans la table "Bien"
SELECT COUNT(*) AS "Nombre de lignes" FROM Bien;

-- Nombre d'enregistrements dans la table "Commune"
SELECT COUNT(*) AS "Nombre de lignes" FROM Commune;

-- Nombre d'enregistrements dans la table "Region"
SELECT COUNT(*) AS "Nombre de lignes" FROM Region;

-- Nombre d'enregistrements dans la table "Population"
SELECT COUNT(*) AS "Nombre de lignes" FROM Population;
```

- Vérification de l’intégrité des données :

```
SELECT COUNT(*) FROM Bien WHERE surface_carrez IS NULL;
```

	COUNT(*)
1	0

```
SELECT COUNT(*) FROM Commune WHERE code_departement IS NULL;
```

	COUNT(*)
1	0

- Vérification des jointures :

```
SELECT b.*, c.*
FROM Bien b
JOIN Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
LIMIT 10;
```

Nombre de lignes chargées : 10

id_bien	id_codedep_cc	no_voie	BTQ	type_voie	voie	surface_carrez	total_piece	surface_local
1	01103	347	NULL	RUE	DU CHATEAU	48,22	3	48

type_local	id_codedep	code_depa	nom_commune	code_region
Appartement	01103	1	Chevry	84

Requêtes SQL et résultats

Analyse des données via SQL

Les requêtes SQL permettent d’extraire, manipuler et analyser les données stockées dans une base. Elles servent à filtrer, trier, agréger et vérifier la cohérence des informations, facilitant ainsi l’exploitation des données pour la prise de décision.

Requête 1

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

Requête :

```
SELECT COUNT(*) AS total_appartements_vendus_1er_semestre_2020
FROM Vente
JOIN Bien ON Vente.id_bien = Bien.id_bien
WHERE Bien.type_local = 'Appartement';
```

Résultat :

	total appartements vendus 1er semestre 2020
1	31378

Requête 2

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

Requête :

```sql
SELECT
    r.nom_region,
    COUNT(v.id_vente) AS nombre_ventes_appartements
FROM
    Vente v
JOIN
    Bien b ON v.id_bien = b.id_bien
JOIN
    Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
JOIN
    Region r ON c.code_region = r.code_region
WHERE
    b.type_local = 'Appartement'
    AND v.date BETWEEN '01/01/2020' AND '30/06/2020'
GROUP BY
    r.nom_region
ORDER BY
    nombre_ventes_appartements DESC;
```

Requête 2 (suite)

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

Résultat :

	nom_region	nombre_ventes_appartements
1	Ile-de-France	13693
2	Provence-Alpes-Côte d'Azur	3583
3	Auvergne-Rhône-Alpes	3166
4	Nouvelle-Aquitaine	1885
5	Occitanie	1617
6	Pays de la Loire	1326
7	Hauts-de-France	1235
8	Grand Est	967
9	Bretagne	960
10	Normandie	847
11	Centre-Val de Loire	684
12	Bourgogne-Franche-Comté	373
13	Corse	217
14	Martinique	91
15	La Réunion	44
16	Guyane	31
17	Guadeloupe	1

Requête 3

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

Requête :

```
SELECT
    b.total_piece,
    COUNT(b.id_bien) AS nombre_ventes,
    ROUND((COUNT(b.id_bien) * 100.0 /
           (SELECT COUNT(*)
            FROM Vente v
            JOIN Bien b2 ON v.id_bien = b2.id_bien
            WHERE b2.type_local = 'Appartement')), 2) AS proportion_ventes
FROM
    Bien b
JOIN
    Vente v ON b.id_bien = v.id_bien
WHERE
    b.type_local = 'Appartement'
GROUP BY
    b.total_piece
ORDER BY
   total_piece ASC;
```

Requête 3 (suite)

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

Résultat :

	total piece	nombre ventes	proportion ventes
1	0	30	0.1
2	1	6739	21.48
3	2	9783	31.18
4	3	8966	28.57
5	4	4460	14.21
6	5	1114	3.55
7	6	204	0.65
8	7	54	0.17
9	8	17	0.05
10	9	8	0.03
11	10	2	0.01
12	11	1	0

Requête 4

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

Requête :

```
SELECT
    c.code_departement,
    ROUND(AVG(v.valeur / b.surface_local)) AS prix_m2_moyen
FROM
    Vente v
JOIN
    Bien b ON v.id_bien = b.id_bien
JOIN
    Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
WHERE
    b.surface_local > 0
GROUP BY
    c.code_departement
ORDER BY
    prix_m2_moyen DESC
LIMIT 10;
```

Requête 4 (suite)

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

Résultat :

	code departement	prix m2 moyen
1	75	12129
2	92	7415
3	94	5398
4	6	4685
5	93	4371
6	74	4149
7	78	4126
8	69	4063
9	2A	3921
10	33	3838

Requête 5

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

Requête :

```sql
SELECT
    ROUND(AVG(v.valeur / b.surface_local)) AS prix_moyen_m2_maison_IDF
FROM
    Vente v
JOIN
    Bien b ON v.id_bien = b.id_bien
JOIN
    Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
JOIN
    Region r ON c.code_region = r.code_region
WHERE
    r.code_region = 11
    AND b.type_local = 'Maison'
    AND b.surface_local > 0;
```

Résultat :

	prix_moyen_m2_maison_IDF
1	3997

Requête 6

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

Requête :

```
SELECT
    v.id_bien AS id_bien,
    v.valeur AS valeur_vente,
    r.nom_region AS region,
    b.surface_local AS surface_local
FROM vente v
JOIN bien b ON v.id_bien = b.id_bien
JOIN commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
JOIN region r ON c.code_region = r.code_region
WHERE b.type_local = 'Appartement'
ORDER BY valeur_vente DESC
LIMIT 10;
```

Requête 6 (suite)

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

Résultat :

	id bien	valeur vente	region	surface local
1	30603	9000000	Ile-de-France	10
2	5261	8600000	Ile-de-France	62
3	3625	8577713	Ile-de-France	289
4	7602	7620000	Ile-de-France	42
5	9988	7600000	Ile-de-France	200
6	17823	7535000	Ile-de-France	143
7	410	7420000	Ile-de-France	357
8	16357	7200000	Ile-de-France	241
9	1924	7050000	Ile-de-France	310
10	19161	6600000	Ile-de-France	76

Requête 7

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

Requête :

```
WITH ventes_t1_2020 AS (
    SELECT COUNT(v.id_vente) AS nb_ventes_t1_2020
    FROM vente v
    WHERE
        DATE(SUBSTR(v.date, 7, 4) || '-' || SUBSTR(v.date, 4, 2) || '-' || SUBSTR(v.date, 1, 2))
        BETWEEN '2020-01-01' AND '2020-03-31'
),

ventes_t2_2020 AS (
    SELECT COUNT(v.id_vente) AS nb_ventes_t2_2020
    FROM vente v
    WHERE
        DATE(SUBSTR(v.date, 7, 4) || '-' || SUBSTR(v.date, 4, 2) || '-' || SUBSTR(v.date, 1, 2))
        BETWEEN '2020-04-01' AND '2020-06-30'
)
```

Requête 7 (suite)

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

```
SELECT
    ventes_t1_2020.nb_ventes_t1_2020 AS nombre_ventes_T1_2020,
    ventes_t2_2020.nb_ventes_t2_2020 AS nombre_ventes_T2_2020,

    FORMAT('%.1f%%',
        CAST(
            (ventes_t2_2020.nb_ventes_t2_2020 - ventes_t1_2020.nb_ventes_t1_2020)
            AS FLOAT
        ) / ventes_t1_2020.nb_ventes_t1_2020 * 100
    ) AS evolution_T2_vs_T1
FROM ventes_t1_2020, ventes_t2_2020;
```

Résultat :

	nombre_ventes_T1_2020	nombre_ventes_T2_2020	evolution_T2_vs_T1
1	16776	17393	3.7%

Requête 8

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

Requête :

```
SELECT
    r.nom_region,
    ROUND(SUM(v.valeur) / SUM(b.surface_local), 2) AS prix_m2_moyen
FROM Vente v
JOIN Bien b ON v.id_bien = b.id_bien
JOIN Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
JOIN Region r ON c.code_region = r.code_region
WHERE b.total_piece > 4
GROUP BY r.nom_region
ORDER BY prix_m2_moyen DESC;
```

Résultat :

	nom_region	prix_m2_moyen
1	Ile-de-France	7055
2	Corse	4809
3	La Réunion	3484
4	Provence-Alpes-Côte d'Azur	3297
5	Martinique	3070
6	Auvergne-Rhône-Alpes	2988
7	Nouvelle-Aquitaine	2933
8	Pays de la Loire	2427
9	Bretagne	2234
10	Occitanie	2201
11	Hauts-de-France	2181
12	Normandie	2122
13	Centre-Val de Loire	1615
14	Grand Est	1465
15	Bourgogne-Franche-Comté	1154

Requête 9

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

Requête :

```
SELECT c.nom_commune AS commune,
       COUNT(v.id_vente) AS nombre_de_ventes_T1
FROM vente v
JOIN bien b ON v.id_bien = b.id_bien
JOIN commune c ON b.id_codedep_codecommune = c.id_codedep_codecommun
WHERE (SUBSTR(v.date, 7, 4) || '-' || SUBSTR(v.date, 4, 2) || '-' || SUBSTR(v.date, 1, 2))
      BETWEEN '2020-01-01' AND '2020-03-31'
GROUP BY c.nom_commune
HAVING nombre_de_ventes_T1 >= 50
ORDER BY nombre_de_ventes_T1 DESC;
```

Requête 9 (suite)

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

Résultat :

	commune	nombre_de_ventes_T1
1	Paris 17e	228
2	Paris 15e	215
3	Paris 18e	209
4	Nice	173
5	Paris 11e	169
6	Paris 16e	165
7	Bordeaux	157
8	Paris 14e	146
9	Paris 20e	127
10	Nantes	119
11	Paris 19e	116
12	Paris 12e	110
13	Paris 10e	109
14	Paris 9e	106
15	Grenoble	106
16	Boulogne-Billancourt	99
17	Paris 13e	94
18	Paris 7e	87
19	Paris 6e	86
20	Marseille 8e	81
21	Asnières-sur-Seine	81
22	Courbevoie	80
23	Paris 5e	79
24	Paris 3e	79
25	Toulouse	78
26	Antibes	77
27	Marseille 4e	72
28	Marseille 1er	71
29	Vincennes	68
30	Rueil-Malmaison	68
31	Lille	67
32	Marseille 9e	66
33	Montreuil	65
34	Angers	64
35	Nîmes	63
36	Sète	62
37	Paris 8e	62
38	La Ciotat	62
39	Rennes	61
40	Paris 2e	61
41	Paris 4e	60
42	Toulon	59
43	Levallois-Perret	59
44	Saint-Maur-des-Fossés	56
45	Versailles	54
46	Ajaccio	54
47	Puteaux	53
48	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.

Requête :

```
WITH prix_m2 AS (

    SELECT
        b.total_piece,
        ROUND(AVG(v.valeur / b.surface_local), 2) AS prix_m2_moyen
    FROM Vente v
    JOIN Bien b ON v.id_bien = b.id_bien
    WHERE b.type_local = 'Appartement'
    AND b.total_piece IN (2, 3)
    GROUP BY b.total_piece
)
```

```
SELECT
    ROUND((SELECT prix_m2_moyen FROM prix_m2 WHERE total_piece = 2), 2) || ' €/m²' AS prix_m2_2p,
    ROUND((SELECT prix_m2_moyen FROM prix_m2 WHERE total_piece = 3), 2) || ' €/m²' AS prix_m2_3p,
    ROUND(
        ((SELECT prix_m2_moyen FROM prix_m2 WHERE total_piece = 3) -
         (SELECT prix_m2_moyen FROM prix_m2 WHERE total_piece = 2)) /
         (SELECT prix_m2_moyen FROM prix_m2 WHERE total_piece = 2) * 100,
    2) || ' %' AS difference_pourcentage;
```

Résultat :

	prix_m2_2p	prix_m2_3p	difference_pourcentage
1	4932.42 €/m²	4284.92 €/m²	-13.13 %

Requête 11

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

Requête :

```
WITH RankedCommunes AS (
    SELECT
        c.nom_commune AS commune,
        c.code_departement,
        ROUND(AVG(v.valeur), 0) AS valeur_fonciere_moyenne,
        ROW_NUMBER() OVER (PARTITION BY c.code_departement ORDER BY AVG(v.valeur) DESC) AS rank
    FROM
        Vente v
    JOIN
        Bien b ON v.id_bien = b.id_bien
    JOIN
        Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
    WHERE
        c.code_departement IN ('6', '13', '33', '59', '69')
        AND v.valeur <> ''
    GROUP BY
        c.nom_commune, c.code_departement
)
```

Requête 11 (suite)

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

```
SELECT
    rc.code_departement,
    rc.commune,
    rc.valeur_fonciere_moyenne
FROM
    RankedCommunes rc
WHERE
    rc.rank <= 3
ORDER BY
    rc.code_departement, rc.valeur_fonciere_moyenne DESC;
```

Requête 11 (suite)

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

Résultat :

	code_departement	commune	valeur_fonciere_moyenne
1	13	Gignac-la-Nerthe	330000
2	13	Saint-Savournin	314425
3	13	Cassis	313417
4	33	Lège-Cap-Ferret	549501
5	33	Vayres	335000
6	33	Arcachon	307436
7	59	Bersée	433202
8	59	Cysoing	408550
9	59	Halluin	322250
10	6	Saint-Jean-Cap-Ferrat	968750
11	6	Eze	655000
12	6	Mouans-Sartoux	476898
13	69	Ville-sur-Jarnioux	485300
14	69	Lyon 2e	455217
15	69	Lyon 6e	426968

Requête 12

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

Requête :

```
SELECT
    c.nom_commune AS commune,
    ROUND((COUNT(v.id_vente) * 1000.0) / p.population_totale, 2) AS transactions_pour_1000_hab
FROM Vente v
JOIN Bien b ON v.id_bien = b.id_bien
JOIN Commune c ON b.id_codedep_codecommune = c.id_codedep_codecommune
JOIN Population p ON c.id_codedep_codecommune = p.id_codedep_codecommune
WHERE p.population_totale > 10000
GROUP BY c.nom_commune
ORDER BY transactions_pour_1000_hab DESC
LIMIT 20;
```

Requête 12 (suite)

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

Résultat :

	commune	transactions_pour_1000_hab
1	Paris 2e	5.84
2	Paris 1er	4.92
3	Paris 3e	4.69
4	Arcachon	4.62
5	La Baule-Escoublac	4.58
6	Paris 4e	4.08
7	Roquebrune-Cap-Martin	3.99
8	Paris 8e	3.83
9	Sanary-sur-Mer	3.5
10	Paris 9e	3.43
11	La Londe-les-Maures	3.43
12	Paris 6e	3.38
13	Saint-Cyr-sur-Mer	3.24
14	Chantilly	3.13
15	Saint-Mandé	3.06
16	Pornichet	3.06
17	Paris 10e	3.04
18	Menton	2.94
19	Saint-Hilaire-de-Riez	2.87
20	Vincennes	2.81

Merci !