🏠 Base de Données SQL - Analyse Marché Immobilier France

🎯 Contexte du projet

DATAImmo est une startup spécialisée dans l'analyse du marché immobilier français. L'entreprise exploite les données publiques de la Direction Générale des Finances Publiques (DGFiP) issues du fichier **Demande de Valeurs Foncières (DVF)**, qui recense l'ensemble des mutations immobilières à titre onéreux en France.

Dans le cadre de ce projet académique (OpenClassrooms), j'ai conçu et implémenté une **base de données relationnelle SQL** pour structurer, nettoyer et exploiter ces données brutes, puis répondre à 12 analyses stratégiques demandées par la direction générale.

🚀 Problématique métier

L'équipe DATAImmo dispose d'un **volume massif de données brutes** (fichiers CSV de transactions immobilières) mais rencontre plusieurs obstacles :

- **Données non structurées** : fichiers plats sans relations, redondances, incohérences
- **Absence de modèle normalisé** : impossible d'interroger efficacement les données
- **Besoins analytiques variés** : comparaisons régionales, évolutions temporelles, segmentations par typologie de biens
- **Scalabilité limitée** : nécessité d'un système évolutif pour intégrer les années suivantes

📋 Objectifs de la mission

Concevoir une **architecture de données robuste** permettant de :

1. **Normaliser** les données DVF en schéma relationnel 3NF (Troisième Forme Normale)
2. **Créer un dictionnaire de données** exhaustif documentant chaque table et attribut
3. **Implémenter la base SQL** sous SQLite avec respect des contraintes d'intégrité
4. **Développer 12 requêtes SQL avancées** répondant aux besoins analytiques de la direction

🔧 Périmètre et spécifications techniques

Données sources

- **Source** : Fichier DVF 2020 (données publiques DGFiP)

- **Période** : 1er semestre 2020 (Proof of Concept avant généralisation)
- **Périmètre géographique** : France métropolitaine hors Bas-Rhin, Haut-Rhin, Moselle et Mayotte
- **Format initial** : CSV avec 43 colonnes et millions de lignes

Modélisation conceptuelle

J'ai conçu un **schéma relationnel normalisé** comportant :

- **Tables de dimensions** : Communes, Départements, Régions, Types de biens, Natures de culture
- **Tables de faits** : Mutations, Biens, Parcelles
- **Clés primaires/étrangères** : respect de l'intégrité référentielle
- **Conformité RGPD** : anonymisation des données sensibles

Typologie des biens analysés

Type de bien	Code	Caractéristiques
Maison	1	Habitation individuelle
Appartement	2	Lot de copropriété avec surface Carrez
Dépendance	3	Garage, cave, cellier (isolés)
Local commercial/industriel	4	Bureaux, commerces, entrepôts

🛠️ Outils et compétences mobilisés

- **SQL (SQLite)** : Création BDD, requêtes avancées (JOINtures, CTEs, agrégations, fenêtres)
- **Modélisation de données** : Schéma entité-association, normalisation 3NF
- **Power Query / Excel** : Nettoyage et transformation des données CSV
- **Documentation** : Dictionnaire de données, schéma relationnel
- **Rigueur analytique** : Validation cohérence des résultats, gestion valeurs aberrantes

📊 Livrables

1. Modèle de données

- **Schéma relationnel normalisé** (diagramme entité-association)
- **Dictionnaire de données** complet (43 attributs documentés)
- **Script SQL de création** (tables, contraintes, index)

2. Base de données implémentée

- **Fichier SQLite (.db)** peuplé avec données S1 2020
- **Procédure d'import** documentée (Power Query → SQL)
- **Gestion qualité** : contrôles de cohérence post-import

3. Requêtes SQL analytiques (12 analyses)

#	Analyse demandée	Compétences SQL mises en œuvre
1	Nombre total d'appartements vendus S1 2020	`COUNT()`, `WHERE` type bien
2	Ventes d'appartements par région	`GROUP BY`, `JOIN` géographique
3	Proportion ventes par nombre de pièces	`GROUP BY` pièces, calcul %
4	TOP 10 départements prix/m² le plus élevé	`ORDER BY`, `LIMIT`, agrégation
5	Prix moyen m² maison Île-de-France	`AVG()`, filtre région
6	TOP 10 appartements les plus chers	`ORDER BY DESC`, multi-jointures
7	Évolution Q1 → Q2 2020	`CTE`, calcul taux variation
8	Classement régions prix/m² apparts 4+ pièces	`HAVING`, filtres multiples
9	Communes avec 50+ ventes T1	`GROUP BY`, seuil transactions
10	Écart % prix/m² 2 pièces vs 3 pièces	Sous-requêtes, calcul différentiel
11	Moyennes valeurs foncières TOP 3 communes (5 dpts)	`ROW_NUMBER()`, `PARTITION BY`
12	TOP 20 communes transactions/1000 hab (>10k hab)	Jointure démographique, ratio

4. Présentation structurée

- **Support PowerPoint** : choix de modélisation, résultats clés
- **Auto-évaluation** : documentation méthodologie et difficultés rencontrées

💡 Défis techniques relevés

1. Nettoyage des données

- **Valeurs manquantes** : traitement des surfaces nulles, dates incohérentes
- **Doublons** : identification mutations multiples sur même bien
- **Normalisation géographique** : correspondance codes commune INSEE ↔ libellés

2. Modélisation avancée

- **Gestion lots/locaux** : distinction notion juridique (lots de copropriété) vs fiscale (locaux)
- **Parcelles multi-natures** : une parcelle peut avoir plusieurs natures de culture (sol, jardin)
- **Mutations complexes** : un acte peut comporter plusieurs dispositions (ventes)

3. Optimisation des requêtes

- **Index stratégiques** : sur colonnes de jointure et filtrage fréquent
- **CTEs (Common Table Expressions)** : décomposition requêtes complexes
- **Fonctions fenêtres** : classements et agrégations avancées

🎨 Exemple de requête complexe

Requête #12 : *TOP 20 communes avec le plus de transactions pour 1000 habitants (>10k hab)*

```
WITH transactions_par_commune AS (
    SELECT
        c.code_commune,
        c.nom_commune,
        c.population,
        COUNT(DISTINCT m.id_mutation) AS nb_transactions
    FROM Mutations m
    JOIN Biens b ON m.id_mutation = b.id_mutation
    JOIN Communes c ON b.code_commune = c.code_commune
    WHERE c.population > 10000
    GROUP BY c.code_commune, c.nom_commune, c.population
)
SELECT
    nom_commune,
    population,
    nb_transactions,
    ROUND((nb_transactions * 1000.0) / population, 2) AS tx_pour_1000_hab
FROM transactions_par_commune
ORDER BY tx_pour_1000_hab DESC
LIMIT 20;
```

Pourquoi cette approche est efficiente :
- **CTE** : sépare l'agrégation du calcul de ratio (lisibilité)
- **Filtre population** : appliqué avant agrégation (performance)
- **Calcul ratio normalisé** : permet comparaison communes différentes tailles
- **Tri + limite** : extraction directe du TOP 20

📈 Résultats et insights clés

Les analyses SQL ont permis de dégager :
- **Disparités régionales** : prix/m² x5 entre régions (Île-de-France vs zones rurales)
- **Impact COVID** : baisse -12% transactions Q1→Q2 2020
- **Segmentation typologie** : appartements 2 pièces = 45% du marché
- **Dynamisme local** : corrélation forte entre densité urbaine et volume transactions

🔗 Structure des fichiers
```
📁 Projet-SQL-Immobilier/
├── 📄 README.md (ce fichier)
├── 📁 data/
│   └── dvf_s1_2020_raw.csv
├── 📁 schema/
│   ├── schema_relationnel.pdf
│   └── dictionnaire_donnees.xlsx
├── 📁 sql/
│   ├── 01_create_tables.sql
│   ├── 02_import_data.sql
│   └── 03_requetes_analytiques.sql
├── 📁 docs/
│   ├── notice_dvf.pdf
│   └── presentation.pptx
└── 📄 dataimmo.db (base SQLite)
```

💼 Valeur ajoutée professionnelle

Ce projet démontre ma capacité à :

- **Concevoir des architectures de données scalables** adaptées aux besoins métier
- **Maîtriser SQL de A à Z** : du DDL (Data Definition Language) au DQL (Data Query Language) avancé
- **Nettoyer et transformer des données complexes** issues de sources publiques
- **Documenter rigoureusement** : schémas, dictionnaires, méthodologies
- **Traduire des besoins analytiques** en requêtes SQL optimisées

---

🔗 Liens utiles

- Données DVF - data.gouv.fr
- Documentation SQLite