Requêtes du projet DataImmo

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

```
-- Objectif : Compter le nombre de ventes uniques d'appartements au S1 2020
-- On utilise COUNT DISTINCT sur un triplet pour éviter de compter plusieurs fois la même mutation
SELECT COUNT(DISTINCT v.valeur_fonciere, v.date_mutation, b.code_insee) AS nb_appartements
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
-- Filtrage sur le type 'Appartement' via une sous-requête sur la table de référence
Where b.id_type_local In (Select id_type_local from type_local where libelle_type_local Like 'Appartement%')
AND v.date_mutation BETWEEN '2020-01-01' AND '2020-06-30';
```

Requête 2 : Le nombre de ventes d'appartement par région pour le 1er semestre 2020

```
-- Objectif : Répartition du volume de ventes d'appartements par zone géographique (Région)
-- Note : Cette requête calcule aussi le pourcentage par rapport au total grâce à une Window Function (OVER)
SELECT
 b.nombre_pieces_principales,
 COUNT(*) AS nb_ventes,
-- Calcul de la proportion sur le total des ventes du semestre
 ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER(), 2) AS pourcentage
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
Where b.id_type_local In (Select id_type_local from type_local where libelle_type_local Like 'Appartement%')
```

```
AND v.date_mutation BETWEEN '2020-01-01' AND '2020-06-30'
AND b.nombre_pieces_principales IS NOT NULL
GROUP BY b.nombre_pieces_principales
ORDER BY b.nombre_pieces_principales;
```

Requête 3 : Proportion des ventes d'appartements par le nombre de pièces.

```
-- Objectif : Analyser la distribution des ventes selon la taille des appartements (T1, T2, T3...)
-- On calcule le volume par catégorie et sa part relative (%) sur l'ensemble du semestre
SELECT
b.nombre_pieces_principales,
COUNT(*) AS nb_ventes,
-- Window Function OVER() : permet de diviser le compte local par le total global sans faire de sous-requête
ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER(), 2) AS pourcentage
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
WHERE b.id_type_local = 2 -- 2 = Appartements
AND v.date_mutation BETWEEN '2020-01-01' AND '2020-06-30'
AND b.nombre_pieces_principales IS NOT NULL
GROUP BY b.nombre_pieces_principales
ORDER BY b.nombre_pieces_principales;
```

Requête 4 : Liste des 10 départements où le prix du mètre carré est le plus élevé.

```
-- Objectif : Identifier les 10 départements où le prix moyen au m² est le plus élevé
-- La jointure utilise LEFT(LPAD...) pour extraire proprement le code département du code INSEE
SELECT
d.dep_nom,
CASE
    WHEN TRIM(b.code_insee) LIKE '97%' THEN LEFT(TRIM(b.code_insee), 3)
    ELSE LEFT(TRIM(b.code_insee), 2)
END AS code_dep,
-- Moyenne du ratio Valeur / Surface pour obtenir le prix au m²
ROUND(AVG(v.valeur_fonciere / b.surface_reelle_bati), 2) AS prix_m2_moyen
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
-- Reconstruction du code département (gestion des codes à 1 chiffre avec LPAD)
JOIN departements d ON d.dep_code = (
  CASE
    WHEN TRIM(b.code_insee) LIKE '97%' THEN LEFT(TRIM(b.code_insee), 3)
    ELSE LEFT(TRIM(b.code_insee), 2)
  END
)
WHERE v.date_mutation BETWEEN '2020-01-01' AND '2020-06-30'
-- Exclusion des surfaces et valeur foncières trop petites pour le calcul
AND b.surface_reelle_bati > 0
AND v.valeur_fonciere > 0
GROUP BY d.dep_nom, code_dep
ORDER BY prix_m2_moyen DESC
LIMIT 10;
```

Requête 5 : Prix moyen du mètre carré d'une maison en Île-de-France.

```
-- Objectif : Obtenir un indicateur de prix spécifique pour les maisons en IDF
-- Jointure en cascade : Bien -> Commune -> Département -> Région pour filtrer géographiquement
SELECT
ROUND(AVG(v.valeur_fonciere / b.surface_reelle_bati), 2) AS prix_m2_moyen_maison_IDF
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
JOIN departements d ON d.dep_code = (
  CASE
    WHEN TRIM(b.code_insee) LIKE '97%' THEN LEFT(TRIM(b.code_insee), 3)
    ELSE LEFT(TRIM(b.code_insee), 2)
  END
)
-- Filtres : Type 'Maison' et Code Région '11' (Île-de-France), temporel : 1er semestre 2020 et on exclus les valeurs nulles pour éviter la division par zéro ou surfaces erronées.
Where b.id_type_local In (Select id_type_local from type_local where libelle_type_local Like 'Maison%')
 AND v.date_mutation BETWEEN '2020-01-01' AND '2020-06-30'
 AND b.surface_reelle_bati > 10
 AND v.valeur_fonciere > 10
AND d.reg_code = '11';
```

Requete 6 : Liste des 10 appartements les plus chers avec la région et le nombre de mètres carrés.

```
-- Objectif : Lister les 10 transactions les plus importantes pour des appartements
-- Inclut la transformation des codes régions en noms via un CASE pour le rapport final
SELECT
v.valeur_fonciere,
b.surface_reelle_bati AS surface_m2,
-- Traduction des codes régions en clair pour la présentation et gestion d'exceptions
  CASE
    WHEN d.reg_code = '11' THEN 'Île-de-France'
    WHEN d.reg_code = '93' THEN 'Provence-Alpes-Côte d\'Azur'
    WHEN d.reg_code = '84' THEN 'Auvergne-Rhône-Alpes'
    WHEN d.reg_code = '76' THEN 'Occitanie'
    WHEN d.reg_code = '75' THEN 'Nouvelle-Aquitaine'
    ELSE CONCAT('Région ', d.reg_code)
  END AS region_nom, b.code_insee
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
-- Jointure adaptative pour gérer les départements métropolitains (2 chiffres) et DOM (3 chiffres)
JOIN departements d ON d.dep_code = (
  CASE
    WHEN TRIM(b.code_insee) LIKE '97%' THEN LEFT(TRIM(b.code_insee), 3)
    ELSE LEFT(TRIM(b.code_insee), 2)
  END
)
Where b.id_type_local In (Select id_type_local from type_local where libelle_type_local Like
'Appartement%')
and b.surface_reelle_bati >10
ORDER BY v.valeur_fonciere DESC
LIMIT 10;
```

Requete 7 : Taux d’évolution du nombre de ventes entre le premier et le second trimestre de 2020.

```
-- Objectif : Mesurer l'impact du contexte (ex : confinement) sur le volume des transactions
-- On utilise deux CTE pour isoler le nombre de ventes de chaque trimestre
WITH stats_trimestres AS (
SELECT
SUM(CASE WHEN MONTH(date_mutation) BETWEEN 1 AND 3 THEN 1 ELSE 0 END) AS ventes_Q1,
SUM(CASE WHEN MONTH(date_mutation) BETWEEN 4 AND 6 THEN 1 ELSE 0 END) AS ventes_Q2
FROM ventes
WHERE YEAR(date_mutation) = 2020
)
-- Calcul du taux de croissance : ((V2 - V1) / V1) * 100
SELECT
ventes_Q1,
ventes_Q2,
ROUND(((ventes_Q2 - ventes_Q1) / ventes_Q1) * 100, 2) AS taux_evolution_pourcentage
FROM stats_trimestres;
```

Requete 8 : Le classement des régions par rapport au prix au mètre carré des appartement de plus de 4 pièces.

```
-- Objectif : Identifier les régions les plus chères pour les "grands" appartements (familles)
-- La jointure s'arrête à la table 'regions' pour permettre le regroupement (GROUP BY)
SELECT
nom_region,
ROUND(AVG(valeur_fonciere / surface_reelle_bati), 2) AS prix_m2_moyen
FROM vue_ventes_regions
-- Filtres : Grands appartements uniquement
WHERE b.id_type_local IN (SELECT id_type_local FROM type_local WHERE libelle_type_local LIKE 'Appartement%')
AND nombre_pieces_principales > 4
AND surface_reelle_bati > 10
GROUP BY nom_region
ORDER BY prix_m2_moyen DESC;
```

Requête 9 : Liste des communes ayant eu au moins 50 ventes au 1er trimestre

```
-- Objectif : Repérer les marchés immobiliers les plus dynamiques (fort volume de transactions)
SELECT
c.nom_commune, b.code_insee,
COUNT(v.id_vente) AS nombre_de_ventes
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
JOIN communes c ON b.code_insee = c.code_insee
-- Période restreinte au Trimestre 1
WHERE v.date_mutation BETWEEN '2020-01-01' AND '2020-03-31'
GROUP BY c.nom_commune, b.code_insee
-- HAVING est utilisé ici (et non WHERE) car on filtre sur le résultat d'un agrégat (COUNT)
HAVING nombre_de_ventes >= 50
ORDER BY nombre_de_ventes DESC;
```

Requete 10 : Différence en pourcentage du prix au mètre carré entre un appartement de 2 pièces et un appartement de 3 pièces.

```
-- Objectif : Comparer le prix moyen au m² entre un appartement de 2 pièces et de 3 pièces
-- Utilisation de Common Table Expressions (WITH) pour isoler les deux calculs

WITH prix_T2 AS (
SELECT AVG(v.valeur_fonciere / b.surface_reelle_bati) AS avg_t2
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
Where b.id_type_local In (Select id_type_local from type_local where libelle_type_local Like 'Appartement%')
AND b.nombre_pieces_principales = 2
AND b.surface_reelle_bati > 10
AND v.valeur_fonciere > 0
),
prix_T3 AS (
SELECT AVG(v.valeur_fonciere / b.surface_reelle_bati) AS avg_t3
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
Where b.id_type_local In (Select id_type_local from type_local where libelle_type_local Like 'Appartement%')
AND b.nombre_pieces_principales = 3
AND b.surface_reelle_bati > 10
AND v.valeur_fonciere > 0
)
-- Calcul final du pourcentage de différence
SELECT
ROUND(avg_t2, 2) AS prix_m2_T2,
ROUND(avg_t3, 2) AS prix_m2_T3,
ROUND(((avg_t2 - avg_t3) / avg_t3) * 100, 2) AS difference_pourcentage
FROM prix_T2, prix_T3;
```

Requete 11 : Les moyennes de valeurs foncières pour le top 3 des communes des départements 6, 13, 33, 59 et 69.

```
-- Objectif : Pour une liste de départements cibles, trouver les 3 communes ayant les plus hautes valeurs foncières moyennes

WITH ClassementCommunes AS (
SELECT
d.dep_nom,
c.nom_commune,
AVG(v.valeur_fonciere) AS moyenne_valeur_fonciere,
-- Attribution d'un rang par département selon la moyenne décroissante
DENSE_RANK() OVER (PARTITION BY d.dep_code ORDER BY AVG(v.valeur_fonciere) DESC) as rang
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
JOIN departements d ON d.dep_code = CAST(LEFT(LPAD(TRIM(b.code_insee), 5, '0'), 2) AS UNSIGNED)
JOIN communes c ON b.code_insee = c.code_insee
-- Filtrage sur les départements demandés (06, 13, 33, 59, 69)
WHERE d.dep_code IN (6, 13, 33, 59, 69)
GROUP BY d.dep_nom, d.dep_code, c.nom_commune
)
-- On ne conserve que les communes classées 1, 2 ou 3
SELECT
dep_nom,  nom_commune,
ROUND(moyenne_valeur_fonciere, 2) AS valeur_moyenne
FROM ClassementCommunes
WHERE rang <= 3
ORDER BY dep_nom, valeur_moyenne DESC;
```

Requete 12 : Les 20 communes avec le plus de transactions pour 1000 habitants pour les communes qui dépassent les 10 000 habitants.

```
SELECT
c.nom_commune,
c.population,
COUNT(v.id_vente) AS nb_ventes,
ROUND((COUNT(v.id_vente) / c.population) * 1000, 2) AS ventes_pour_1000_hab
FROM ventes v
JOIN biens b ON v.id_bien = b.id_bien
JOIN communes c ON b.code_insee = c.code_insee
WHERE c.population > 10000
GROUP BY c.nom_commune, c.population
ORDER BY ventes_pour_1000_hab DESC
LIMIT 20;
```