Les données foncières (DVF)

- Vedène -
- Exemple sur fulldvf France de 2020 à 2025
- Etude Bandol - apprentissage de renseignement des indicateurs

Vedène -

C'est un excellent test pour valider l'industrialisation du processus. Vedène (84141) va nous permettre de vérifier si la structure à 40 colonnes est constante.

Voici le plan d'action séquentiel.

1. Création de `vedene_full` (Le moule brut)

On prépare d'abord le réceptacle pour l'importation du CSV.

SQL

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
CREATE SCHEMA IF NOT EXISTS dvf_raw;

DROP TABLE IF EXISTS dvf_raw.vedene_full;
CREATE TABLE dvf_raw.vedene_full (
    c1 text, c2 text, c3 text, c4 text, c5 text, c6 text, c7 text, c8 text, c9 text, c10 text,
    c11 text, c12 text, c13 text, c14 text, c15 text, c16 text, c17 text, c18 text, c19 text,
c20 text,
    c21 text, c22 text, c23 text, c24 text, c25 text, c26 text, c27 text, c28 text, c29 text,
c30 text,
    c31 text, c32 text, c33 text, c34 text, c35 text, c36 text, c37 text, c38 text, c39 text,
c40 text
);
"
```

2. Injection des données (Bash/Docker)

Assure-toi que le fichier est bien présent dans ton arborescence locale avant de lancer :

Bash

```
# 1. Injection du CSV (40 colonnes)
cat "/home/debian/data/Regions/93/84/248400251/84141/84141.csv" | docker exec -i
alteris_postgis psql -U alteris_admin -d alteris_geo -c "COPY dvf_raw.vedene_full FROM STDIN
WITH (FORMAT CSV, HEADER, DELIMITER ',');"

# 2. Création du Jalon Géographique (Points Lambert 93)
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
DROP TABLE IF EXISTS dvf_raw.vedene_valide;
CREATE TABLE dvf_raw.vedene_valide AS
SELECT
    f.c1 as id_mutation,
    ST_Transform(ST_SetSRID(ST_MakePoint(NULLIF(f.c39, '')::double precision, NULLIF(f.c40,
'')::double precision), 4326), 2154) as geom
FROM dvf_raw.vedene_full f
WHERE f.c39 ~ '^-?[0-9]';
CREATE INDEX sidx_vedene_valide ON dvf_raw.vedene_valide USING GIST (geom);
"
```

3. Création de `vedene_valide` (Le Jalon Géographique)

On extrait les points GPS pour "poser le premier jalon" et vérifier l'emprise.

SQL

```
DROP TABLE IF EXISTS dvf_raw.vedene_valide;
CREATE TABLE dvf_raw.vedene_valide AS
SELECT
    f.c1 as id_mutation,
    ST_Transform(ST_SetSRID(ST_MakePoint(NULLIF(f.c39, '')::double precision, NULLIF(f.c40,
'')::double precision), 4326), 2154) as geom
FROM dvf_raw.vedene_full f
WHERE f.c39 ~ '^-?[0-9]';

CREATE INDEX sidx_vedene_valide ON dvf_raw.vedene_valide USING GIST (geom);
```

4. Fusion Finale : `84141_dvf_alteris` (L'Item Unique)

On assemble le tout (DVF + Géométrie Validée + Jointure Cadastre) en éliminant les doublons de lignes DVF.

SQL

```
DROP TABLE IF EXISTS dvf_raw."84141_dvf_alteris";
CREATE TABLE dvf_raw."84141_dvf_alteris" AS
WITH cadastre AS (
    -- Reconstruction spatiale du cadastre de Vedène
    SELECT
        (f->'properties'->>'id')::text as id_parcelle_cad,
        ST_Transform(ST_SetSRID(ST_GeomFromGeoJSON(f->'geometry'), 4326), 2154) as geom_poly
    FROM (
        SELECT jsonb_array_elements(string_agg(line, '')::jsonb->'features') as f
        FROM dvf_raw.tmp_vedene_txt
    ) sub
)
SELECT DISTINCT ON (f.c1)
    v.geom,
    f.c1 as id_mutation,
    NULLIF(f.c2, '')::date as date_mutation,
    NULLIF(f.c5, '')::numeric as valeur_fonciere,
    f.c16 as id_parcelle_dvf,
    f.c8 || ' ' || f.c9 as adresse,
    f.c28 as type_local,
    NULLIF(f.c29, '')::numeric as surface_bati,
    ST_Within(v.geom, c.geom_poly) as certification_spatiale
FROM dvf_raw.vedene_full f
JOIN dvf_raw.vedene_valide v ON f.c1 = v.id_mutation
LEFT JOIN cadastre c ON f.c16 = c.id_parcelle_cad
ORDER BY f.c1;

-- Nettoyage des tables temporaires
DROP TABLE dvf_raw.tmp_vedene_txt;
```

Chiche ! Les scripts sont prêts. Tu veux que je lance l'exécution ou tu préfères vérifier un point de structure avant ?

Exemple sur fulldvf France de 2020 à 2025

1/ on a regroupé les fichiers fullDVF sur le home/debian/data

2/ on a lancé le script

```
#!/bin/bash
TARGET_TABLE="dvf_raw.var_history_83"
DATA_DIR="/home/debian/data"

echo "-----------------------------------------------------"
echo "ALTERIS - Moissonnage VAR (83) - Colonne 13"
echo "-----------------------------------------------------"

# 1. On vide la table pour repartir sur une base propre
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "TRUNCATE $TARGET_TABLE;"

# 2. Boucle sur les 6 fichiers
for YEAR in {2020..2025}; do
    FILE="$DATA_DIR/fullDVF_${YEAR}.csv"

    if [ -f "$FILE" ]; then
        echo "--> Extraction Var 83 dans l'année $YEAR..."

        # On prépare le fichier temporaire avec l'entête
        head -n 1 "$FILE" > "$DATA_DIR/var_tmp.csv"

        # FILTRE : On cherche exactement le code 83 après 12 virgules
        grep -E "^([^,]*,){12}83," "$FILE" >> "$DATA_DIR/var_tmp.csv"

        # Injection
        cat "$DATA_DIR/var_tmp.csv" | docker exec -i alteris_postgis psql -U alteris_admin -d
alteris_geo -c "COPY $TARGET_TABLE FROM STDIN WITH (FORMAT CSV, HEADER, DELIMITER ',');"

        # Nettoyage
        rm "$DATA_DIR/var_tmp.csv"
    fi
done
```

```
echo "-------------------------------------------------"
echo "RÉSULTAT DE L'OBJECTIVATION :"
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "SELECT count(*) as total_var_83 FROM $TARGET_TABLE;"
```

On monte la table pure

C'est la voix de la sagesse **Alteris** : d'abord figer la donnée brute certifiée (le socle), puis construire la production par-dessus. On ne mélange pas l'archive et l'analyse.

On va donc créer la table `DVFFull83_raw`. Ce sera ta "Banque Centrale" du Var : une table physique, indexée, contenant l'intégralité des 451 336 lignes sans aucune transformation, mais avec une colonne géométrique propre pour QGIS.

1. Bloc d'évolutions proposées

- **Matérialisation :** On passe d'une vue à une **TABLE** physique (plus rapide pour QGIS).
- **Typage SQL :** On convertit les colonnes critiques (Date, Prix, Surface) pour permettre les tris immédiats, mais on garde tout le reste.
- **Indexation Double :** Un index spatial (GIST) pour la carte et un index attributaire sur le `code_commune`.

---

2. Création de la Table de Référence : `DVFFull83_raw`

Lance ce bloc pour graver tes données dans le marbre :

SQL

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
DROP TABLE IF EXISTS dvf_raw.\"DVFFull83_raw\";

CREATE TABLE dvf_raw.\"DVFFull83_raw\" AS
SELECT
    -- Géométrie Lambert 93 générée à la volée
    ST_Transform(ST_SetSRID(ST_MakePoint(NULLIF(c39, '')::double precision, NULLIF(c40, '')::double precision), 4326), 2154) as geom,
    c1 as id_mutation,
```

```
    NULLIF(c2, '')::date as date_mutation,
    c4 as nature_mutation,
    NULLIF(c5, '')::numeric as valeur_fonciere,
    c8 as adresse_numero,
    c9 as adresse_nom_voie,
    c10 as code_postal,
    c11 as code_commune,
    c12 as nom_commune,
    c13 as code_departement,
    c16 as id_parcelle,
    c29 as lot1_surface_carrez,
    c31 as type_local,
    NULLIF(c32, '')::numeric as surface_reelle_bati,
    NULLIF(c33, '')::integer as nombre_pieces_principales,
    NULLIF(c38, '')::numeric as surface_terrain,
    c39 as longitude_wgs84,
    c40 as latitude_wgs84
FROM dvf_raw.var_history_83
WHERE c39 ~ '^-?[0-9]';

-- Indexation de performance
CREATE INDEX sidx_dvffull83_geom ON dvf_raw.\"DVFFull83_raw\" USING GIST (geom);
CREATE INDEX idx_dvffull83_commune ON dvf_raw.\"DVFFull83_raw\" (code_commune);
CREATE INDEX idx_dvffull83_date ON dvf_raw.\"DVFFull83_raw\" (date_mutation);
"
```

---

3. Raisonnement par étape : Pourquoi c'est ta référence ?

1. **Immutabilité :** Cette table contient les 451 336 lignes. Si une vente a 3 lots, tu as les 3 lignes. C'est ta "boîte noire".
2. **Vitesse QGIS :** Contrairement à la table `var_history_83` qui n'a que du texte (TEXT), ici les prix sont des `NUMERIC` et les dates des `DATE`. Les filtres dans QGIS seront instantanés.
3. **Sécurité :** Pour ton rendu de jeudi, si on te pose une question sur un lot spécifique, tu as l'info brute sous la main.

---

4. Audit Final

Une fois la table créée, vérifie que le compte est bon : `SELECT count(*) FROM dvf_raw."DVFFull83_raw";`

C'est fait ? Si oui, tu as officiellement ta base de référence pour le Var. On s'arrête là pour la structure ou tu veux qu'on prépare l'export pour ton NAS ?

L'export en GPKG n'a pas fonctionné - "C'est une erreur classique de "boîte à outils" : l'image Docker `postgis/postgis` contient la base de données, mais elle n'embarque pas toujours les utilitaires de conversion comme `ogr2ogr` (qui appartiennent à la bibliothèque GDAL)."

```
# Export de ta table de référence vers un fichier GeoPackage
docker exec -i alteris_postgis ogr2ogr -f "GPKG" \
/home/debian/data/DVFFull83_raw_2020_2025.gpkg \
PG:"dbname=alteris_geo user=alteris_admin password=alteris_pwd host=localhost" \
"dvf_raw.DVFFull83_raw"
```

Donc export du `DVFFull83_raw`

en csv avec point geolocalisation et ID Cadastre

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
COPY (SELECT * FROM dvf_raw.\"DVFFull83_raw\") TO STDOUT WITH (FORMAT CSV, HEADER, DELIMITER
',');
" > /home/debian/data/DVFFull83_raw_2020_2025.csv
```

Etude Bandol - apprentissage de renseignement des indicateurs

A - Demographie

A3 - l'évolution des CSP par IRIS (la gentrification)

'est la meilleure façon de documenter : garder le code brut sous le coude. Pour l'indicateur **A3 (Profil CSP)**, voici le bloc SQL "propre" que nous avons utilisé.

Il est structuré pour être réutilisable : il suffit de changer les noms des tables ou les codes IRIS pour l'adapter à une autre ville.

□□ Script SQL : Calcul de l'indicateur A3 (Structure des Actifs)

SQL

```
/* INDICATEUR A3 : ANALYSE DE LA POPULATION ACTIVE
   Période : 2016 (CS1) vs 2025 (GS)
   Filtre : Population active uniquement (Codes 2 à 6)
*/

WITH stats_16 AS (
    SELECT
        iris,
```

```
        -- Agrégation par catégorie (Pondération IPONDI)
        SUM(CASE WHEN cs1 = '2' THEN ipondi::numeric ELSE 0 END) as art_16,
        SUM(CASE WHEN cs1 = '3' THEN ipondi::numeric ELSE 0 END) as cad_16,
        SUM(CASE WHEN cs1 = '4' THEN ipondi::numeric ELSE 0 END) as int_16,
        SUM(CASE WHEN cs1 = '5' THEN ipondi::numeric ELSE 0 END) as emp_16,
        SUM(CASE WHEN cs1 = '6' THEN ipondi::numeric ELSE 0 END) as ouv_16,
        -- Total de la population active (Base de calcul pour les %)
        SUM(ipondi::numeric) FILTER (WHERE cs1 IN ('2','3','4','5','6')) as tot_16
    FROM "83009_bandol".stats_individus_2016
    WHERE iris IN ('830090101', '830090102', '830090106')
    GROUP BY iris
),
stats_25 AS (
    SELECT
        iris,
        -- Agrégation par catégorie (Pondération IPONDI)
        SUM(CASE WHEN gs = '2' THEN ipondi::numeric ELSE 0 END) as art_25,
        SUM(CASE WHEN gs = '3' THEN ipondi::numeric ELSE 0 END) as cad_25,
        SUM(CASE WHEN gs = '4' THEN ipondi::numeric ELSE 0 END) as int_25,
        SUM(CASE WHEN gs = '5' THEN ipondi::numeric ELSE 0 END) as emp_25,
        SUM(CASE WHEN gs = '6' THEN ipondi::numeric ELSE 0 END) as ouv_25,
        -- Total de la population active (Base de calcul pour les %)
        SUM(ipondi::numeric) FILTER (WHERE gs IN ('2','3','4','5','6')) as tot_25
    FROM "83009_bandol".stats_individus
    WHERE iris IN ('830090101', '830090102', '830090106')
    GROUP BY iris
)
-- Calcul final des parts et de l'évolution
SELECT
    s25.iris,
    'Cadres' as csp,
    ROUND((s16.cad_16/NULLIF(s16.tot_16,0))*100,1) as part_2016,
    ROUND((s25.cad_25/NULLIF(s25.tot_25,0))*100,1) as part_2025,
    ROUND(((s25.cad_25-s16.cad_16)/NULLIF(s16.cad_16,0))*100,1) as evol_vol_pct
FROM stats_25 s25 JOIN stats_16 s16 ON s25.iris = s16.iris
UNION ALL
SELECT
    s25.iris, 'Prof. Interm.',
    ROUND((s16.int_16/NULLIF(s16.tot_16,0))*100,1),
ROUND((s25.int_25/NULLIF(s25.tot_25,0))*100,1),
    ROUND(((s25.int_25-s16.int_16)/NULLIF(s16.int_16,0))*100,1)
```

```
FROM stats_25 s25 JOIN stats_16 s16 ON s25.iris = s16.iris
ORDER BY iris, part_2025 DESC;
```

□□ Pourquoi avoir codé comme ça ?

- `NULLIF(..., 0)` : C'est la sécurité anti-crash. Si un IRIS n'a aucun cadre en 2016, la division par zéro échoue. Cette fonction transforme le 0 en `NULL` pour éviter l'erreur.
- `FILTER (WHERE ...)` : C'est la méthode SQL moderne pour compter uniquement ce qui nous intéresse dans une somme globale.
- `UNION ALL` : On "empile" les résultats des cadres et des prof. intermédiaires pour avoir une liste facile à lire, plutôt qu'un tableau avec 50 colonnes.

□□ Petit conseil d'expert :

Si tu dois refaire ça pour une autre ville (ex: Sanary), il te suffira de :

1. Changer le préfixe du schéma (`"83009_bandol"`).
2. Changer la liste des codes IRIS dans les clauses `WHERE`.

C'est dans la boîte ! Tu te sens prêt à attaquer la mise en page de ton rapport ou tu as encore un doute sur une colonne ?

C - DVF

Toujours commencer par une approche semantique de la table pour que l'IA ne fasse pas de fautes

debian@vps-02c1f983-vps-ovh-net:~$ docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "SELECT nature_mutation, COUNT(*) FROM \"83009_bandol\".dvf_bandol_complet GROUP BY nature_mutation;"

C5 - Etude du marché de l'ancien

C5a/ - le marche immobilier

Le problème est que les micro-studios "balnéaires" hyper spéculatifs de moins de 20 m² faussent le prix moyen du m² global.

Dans l'analyse C5a - Nexte_stats neutralise ces données pour transcrire un marché immobilier courant incluant les studios d'habitat à l'année, mais excluant les meublés d'été inférieurs à 20 m². (A VALIDER)

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
-- 1. Nettoyage de l'ancienne couche
DROP TABLE IF EXISTS \"83009_bandol\".\"dvf_carto_2022_2025\";
DROP TABLE IF EXISTS \"83009_bandol\".\"dvf_carto_2022_2025_prixm²\";

-- 2. Création de la couche filtrée (> 20m²)
CREATE TABLE \"83009_bandol\".\"dvf_carto_2022_2025_prixm²\" AS
SELECT
    id_mutation,
    MAX(date_mutation) as date_mutation,
    EXTRACT(YEAR FROM MAX(date_mutation)) as annee,
    MAX(type_local) as type_bien,
    MAX(valeur_fonciere) as prix_total,
    SUM(surface_reelle_bati) as surface_totale,
    ROUND(MAX(valeur_fonciere) / NULLIF(SUM(surface_reelle_bati), 0), 0) as prix_m2,
```

```
    ST_Centroid(ST_Collect(geom)) as geom
FROM \"83009_bandol\".dvf_bandol_complet
WHERE nature_mutation IN ('Vente', 'Vente en l''état futur d''achèvement')
  AND type_local IN ('Maison', 'Appartement')
  AND date_mutation >= '2022-01-01'
  AND valeur_fonciere > 50000
GROUP BY id_mutation
HAVING SUM(surface_reelle_bati) >= 20; -- FILTRE ANTI-SINGLETONS

-- 3. Indexation
CREATE INDEX idx_dvf_carto_prixm2_geom ON \"83009_bandol\".\"dvf_carto_2022_2025_prixm²\"
USING GIST(geom);"
```

C5b/ le tableau des typologies globales ancien/neuf en appartements sur les 3 dernières années

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
SELECT
    CASE
        WHEN nombre_pieces_principales = 1 THEN 'Studio'
        WHEN nombre_pieces_principales = 2 THEN 'T2'
        WHEN nombre_pieces_principales = 3 THEN 'T3'
        WHEN nombre_pieces_principales = 4 THEN 'T4'
        ELSE 'T5+'
    END as typologie,
    COUNT(DISTINCT id_mutation) as nb_ventes,
    ROUND(AVG(surface_reelle_bati), 0) as surface_moyenne,
    ROUND(AVG(valeur_fonciere), 0) as prix_moyen,
    ROUND(AVG(valeur_fonciere / NULLIF(surface_reelle_bati, 0)), 0) as prix_m2_moyen
FROM \"83009_bandol\".dvf_bandol_complet
WHERE nature_mutation = 'Vente'
  AND type_local = 'Appartement'
  AND surface_reelle_bati >= 20
  AND date_mutation >= '2022-01-01'
GROUP BY nombre_pieces_principales
ORDER BY nombre_pieces_principales;"
```

C5c - le tableau des typologies globales ancien/neuf en

appartements VEFA sur les 3 dernières années

□□ Documentation Indicateur C5c : Analyse du Neuf (DVF x BDNB)

Cet indicateur permet de filtrer les ventes de la DVF non pas sur le champ "Nature de mutation" (souvent peu fiable), mais sur la réalité physique du bâtiment (Année de construction certifiée par les Fichiers Fonciers).

1. Intégration de la BDNB (Var - 83)

L'import se fait depuis un dump SQL de la Base Nationale des Bâtiments.

Bash

```
# Copie du fichier SQL vers le container
docker cp data/Regions/93/83/bdnb.sql alteris_postgis:/tmp/bdnb.sql

# Restauration dans la base de données
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -f /tmp/bdnb.sql
```

Note : Le schéma créé est `bdnb_2025_07_a_open_data_dep83`.

---

2. Croisement DVF x BDNB (Filtre $\ge$ 2017)

On crée une table de synthèse qui lie les mutations DVF aux caractéristiques du bâtiment via la parcelle cadastrale.

SQL

```sql
-- Création de la table de travail sur le VPS
CREATE TABLE public.temp_bandol_neuf AS
SELECT
    d.id_mutation,
    d.date_mutation,
    d.valeur_fonciere,
    d.surface_reelle_bati,
    d.nombre_pieces_principales,
    ffo.annee_construction,
    d.geom
FROM "83009_bandol".dvf_bandol_complet d
JOIN "bdnb_2025_07_a_open_data_dep83".rel_batiment_groupe_parcelle rel
    ON d.id_parcelle = rel.parcelle_id
JOIN "bdnb_2025_07_a_open_data_dep83".batiment_groupe_ffo_bat ffo
    ON rel.batiment_groupe_id = ffo.batiment_groupe_id
WHERE d.type_local = 'Appartement'
  AND d.surface_reelle_bati > 0
  AND ffo.annee_construction >= 2017;
```

---

3. Export GeoJSON (Projection GPS WGS84)

Pour que le fichier soit immédiatement exploitable dans QGIS sans décalage, on projette la géométrie du Lambert-93 (2154) vers le WGS84 (4326).

Bash

```
# Génération du JSON propre à l'intérieur du container
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
COPY (
    SELECT jsonb_build_object(
        'type',     'FeatureCollection',
        'features', jsonb_agg(features.feature)
    )
    FROM (
      SELECT jsonb_build_object(
        'type',       'Feature',
        'geometry',   ST_AsGeoJSON(ST_Transform(geom, 4326))::jsonb,
        'properties', to_jsonb(inputs) - 'geom'
      ) AS feature
      FROM (SELECT * FROM public.temp_bandol_neuf) inputs
    ) features
) TO '/tmp/final_gps.geojson';"

# Extraction vers le dossier d'échange du VPS
docker cp alteris_postgis:/tmp/final_gps.geojson /home/debian/data/bandol_expert_gps.geojson
```

□□ Résultats obtenus (Bandol)

Le croisement a permis d'identifier **31 transactions** sur le cycle 2017-2025.

Typologie	Nb Ventes	Surface Moy.	Prix m² Moyen
T2	11	44 m²	**7 551 €**
T3	9	68 m²	**7 560 €**
T4	8	91 m²	**6 547 €**

Alerte Point de Vigilance : Une vente record (Studio) à **26 409 €/m²** a été détectée, signalant un segment "Ultra-Luxe" spécifique à isoler de la moyenne générale pour ne pas fausser les prévisionnels de sortie du projet AH0219.

C'est ta "recette de cuisine" pour n'importe quelle autre commune du Var maintenant. On passe à l'analyse par IRIS pour voir si le quartier du projet sur-performe la moyenne de Bandol

Autre aspect C5c - dynamique des VEFA sur Bandol

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "SELECT
CASE         WHEN date_mutation >= '2025-01-01' AND date_mutation <= '2025-06-30' THEN '2025
(S1)'        ELSE TO_CHAR(date_mutation, 'YYYY')     END as periode,    COUNT(DISTINCT
id_mutation) as nb_vefa,    ROUND(AVG(surface_reelle_bati), 0) as surface_moy,
ROUND(AVG(valeur_fonciere / NULLIF(surface_reelle_bati, 0)), 0) as prix_m2_vefaFROM
\"83009_bandol\".dvf_bandol_completWHERE nature_mutation = 'Vente en l''état futur
d''achèvement'  AND type_local = 'Appartement'  AND surface_reelle_bati > 0  AND date_mutation
>= '2022-01-01'GROUP BY periodeORDER BY periode DESC;"
```

C5d - Studios pondérés

C'est une excellente stratégie. En statistiques, on appelle cela une **moyenne tronquée** (ou moyenne élaguée). En excluant les 20 % extrêmes de chaque côté (le "bas de marché" souvent dégradé et les "ovnis" spéculatifs), on obtient l'image la plus fidèle du **cœur de marché**.

Pour être précis : si tes calculs précédents commençaient à 20 m², nous filtrons ici strictement de **12 à 19,99 m²**.

□□ La Requête SQL "Moyenne Tronquée" (Studios 12-19m²)

Voici la commande qui réalise ce filtrage complexe. Elle utilise des "Common Table Expressions" (CTE) pour classer les prix et ne garder que le milieu de la distribution (les 60 % centraux).

```
docker exec -i alteris_postgis psql -U alteris_admin -d alteris_geo -c "
WITH studios_calcules AS (
    SELECT
        valeur_fonciere / surface_reelle_bati as prix_m2,
        surface_reelle_bati,
        valeur_fonciere,
        NTILE(5) OVER (ORDER BY valeur_fonciere / surface_reelle_bati) as quintile
    FROM \"83009_bandol\".dvf_bandol_complet
    WHERE type_local = 'Appartement'
      AND surface_reelle_bati >= 12
      AND surface_reelle_bati < 20
      AND nature_mutation = 'Vente'
      AND date_mutation >= '2022-01-01'
)
SELECT
    COUNT(*) as nb_ventes_retenues,
    ROUND(AVG(surface_reelle_bati), 1) as surface_moyenne,
    ROUND(AVG(prix_m2), 0) as prix_m2_moyen_tronque,
    ROUND(MIN(prix_m2), 0) as borne_basse_retenue,
    ROUND(MAX(prix_m2), 0) as borne_haute_retenue
FROM studios_calcules
WHERE quintile IN (2, 3, 4); -- On exclut le quintile 1 (20% bas) et le 5 (20% haut)"
```

Voici la fiche de documentation technique et méthodologique pour l'indicateur **C5e**. Elle est conçue pour être insérée directement dans

---

□□ Documentation Indicateur C5e : Dynamique des Programmes Immobiliers

Neufs

1. Définition et Objectif

L'indicateur **C5e** vise à isoler et analyser les opérations de promotion immobilière (VEFA) par une approche de **clustering géographique**. Contrairement aux indicateurs globaux, il permet de distinguer la vente au détail (particuliers) de la vente en bloc (institutionnels) et d'identifier les valeurs de sortie réelles par programme.

2. Méthodologie d'Extraction

Le workflow repose sur l'identification de "grappes" de mutations à une même adresse. Le seuil de détection est fixé à :

- **Volume** : Minimum 3 mutations distinctes sur une même voie la même année.
- **Valeur** : Ou un Chiffre d'Affaires cumulé supérieur à 2 000 000 €.

Requête SQL de référence :

SQL

```
SELECT
    adresse_numero AS nom_de_la_rue,
    TO_CHAR(date_mutation, 'YYYY') AS annee,
    COUNT(DISTINCT id_mutation) AS nb_mutations,
    SUM(valeur_fonciere) AS ca_total_dvf
FROM "83009_bandol".dvf_bandol_complet
WHERE nature_mutation LIKE 'Vente%futur%'
  AND date_mutation >= '2022-01-01'
GROUP BY nom_de_la_rue, annee
HAVING COUNT(DISTINCT id_mutation) >= 3
   OR SUM(valeur_fonciere) >= 2000000
ORDER BY annee DESC, ca_total_dvf DESC;
```

---

3. Protocole d'Analyse des Résultats

Une fois la liste des adresses extraite, l'expertise se décompose en trois niveaux de lecture :

Segment détecté	Indicateur DVF	Action Expertise
Vente en Bloc	CA > 20 M€ pour < 5 mutations	Identifier l'acquéreur institutionnel (Bailleur/Foncière) via Sitadel.
Luxe / Exception	Prix moyen / mutation > 1,5 M€	Vérifier la typologie (Grands plateaux, Villas sur toit).
Cœur de Marché	Prix moyen / mutation [400k€ - 900k€]	**Comparable direct** pour les projets de promotion standard.

---

4. Cas Pratique : Analyse de Bandol (2022-2025)

L'application de l'indicateur **C5e** sur la base DVF a permis d'isoler les marqueurs suivants :

- **Pôle Institutionnel (Biais statistique à isoler)** :
  - *Bd de Marseille* (2024) et *Moulin à Vent* (2025) totalisent près de **258 M€** de CA pour seulement 4 mutations majeures. Ces données valident une activité de promotion massive (ventes en bloc) sur la commune.
- **Pôle Comparables Directs (Résidentiel)** :
  - **Rue J.J. Rousseau (2022)** : 10 mutations pour 26,2 M€ (Moyenne 2,6 M€/acte).
  - **Av. du 11 Novembre (2022)** : 8 mutations pour 4,3 M€ (Moyenne **547k€/acte**).
  - **Rue Pasteur (2022)** : 3 mutations pour 2,2 M€ (Moyenne **742k€/acte**).