Jeu de Données

1-La requête sélectionne l'emplacement des logements et compte ceux qui sont disponibles dans chaque zone géographique

Query | Query History

```
238 WITH revenus AS (
239     SELECT
240         tl.type_logement,
241         SUM((r.date_fin - r.date_debut) * l.loyer) AS revenu_total
242     FROM Reservation r
243     JOIN Logement l ON r.id_logement = l.id_logement
244     JOIN Type_logement tl ON l.id_type_logement = tl.id_type_logement
245     GROUP BY tl.type_logement
246 ),
247 interventions AS (
248     SELECT
249         tl.type_logement,
```

Data Output | Messages | Notifications

Showing rows: 1 to 4 | Page No: 1

	type_logement character varying (50)	revenu_total double precision	nombre_interventions bigint	rentabilite_estimee double precision
1	Maison	264500	7	261000
2	Appartement T3	123700	5	121200
3	Studio	53400	4	51400
4	Appartement T2	44550	4	42550

2-Cette requête permet d'identifier les résidents avec des comportements problématiques ou signalés des conflits récurrents:

```sql
SELECT r.nom, r.prenom, COUNT(c.id_conflit) AS nombre_conflits
FROM Resident r
JOIN Resident_conflicts rc ON r.id_resident = rc.id_resident
JOIN Conflit c ON rc.id_conflit = c.id_conflit
WHERE c.resolu != 'oui'
GROUP BY r.id_resident
HAVING COUNT(c.id_conflit) > 1
ORDER BY nombre_conflits DESC;

-- F. Comment organiser les événements communautaires pour maximiser la
```

Data Output | Messages | Notifications

Showing rows: 1 to 5 | Page No

	nom character varying (50)	prenom character varying (50)	nombre_conflits bigint
1	Garcia	Louis	2
2	Petit	Elise	2
3	Michel	Emma	2
4	Thomas	Hugo	2

3-L'objectif de cette requête est de maximiser la participation des résidents aux événements en analysant leur taux de participation.

```sql
SELECT
    Round((COUNT(DISTINCT re.id_resident) * 100.0 / (SELECT COUNT(*) FROM Resident)),2) AS taux_pa
    e.type_evenement,
    COUNT(re.id_resident) AS nombre_participants,

    e.date_event
FROM Evenement e
JOIN participation re ON e.id_evenement = re.id_evenement
WHERE e.date_event BETWEEN '2024-01-01' AND '2025-12-31'  -- Période à ajuster selon vos besoins
GROUP BY e.type_evenement, e.id_evenement, e.date_event
ORDER BY nombre_participants DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 3 | Page No: 1 of 1

	taux_participation numeric	type_evenement character varying (50)	nombre_participants bigint	date_event date
1	16.67	Séminaire résidentiel	2	2025-09-30
2	16.67	Journée nettoyage	2	2025-07-23
3	8.33	Journée nettoyage	1	2024-08-21

4-Cette requête permet de trouver les logements disponibles pendant une période

```
300 SELECT L.id_logement, L.emplacement, L.loyer, T.type_logement
```

Query Query History

```
341 SELECT
342     L.id_logement,
343     L.emplacement,
344     COALESCE(COUNT(I.id_intervention), 0) AS nb_ameliorations,
345     COALESCE(ROUND(AVG(N.score), 2), 0) AS moyenne_notes
346 FROM Logement L
347 LEFT JOIN Logement_Intervention LI ON L.id_logement = LI.id_logement
348 LEFT JOIN Intervention I ON LI.id_intervention = I.id_intervention
349 LEFT JOIN Note N ON L.id_logement = N.id_logement
350 WHERE I.id_type_intervention IN (2, 3)
351 GROUP BY L.id_logement, L.emplacement
352 ORDER BY moyenne_notes DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 9 Page No

	id_logement [PK] integer	emplacement character varying (100)	nb_ameliorations bigint	moyenne_notes numeric
1	3	Rue Nationale, Lyon	2	5.00
2	7	Banlieue	2	5.00
3	5	Quartier résidentiel	2	4.00
4	1	Centre-ville	2	4.00

5-Cette requête génère un profil démographique des résidents en joignant les tables Resident et Profil.

bdrproj/postgres@PostgreSQL 17

No limit

Query Query History

```
406 SELECT
407     r.nom,
408     r.prenom,
409     EXTRACT(YEAR FROM AGE(r.date_naissance)) AS age,
410     p.profession,
411     p.secteur_activite,
412     r.date_entree,
413     r.date_sortie
414 FROM
415     Resident r
416 JOIN
417     Profil p ON r.id_profil = p.id_profil;
```

Data Output Messages Notifications

Showing rows: 1 to 12 Page No: 1 of 1

	nom character varying (50)	prenom character varying (50)	age numeric	profession character varying (100)	secteur_activite character varying (100)	date_entree date	date_sortie date
1	Dupont	Jean	34	Ingénieur	Informatique	2024-03-15	2024-05-20
2	Martin	Sophie	39	Médecin	Santé	2024-06-10	2024-08-25
3	Lefevre	Paul	24	Etudiant	Éducation	2024-02-01	2024-04-15
4	Moreau	Clara	29	Professeur	Éducation	2024-05-05	2024-05-10

6-Cette requête génère le taux d'occupation des logements

```sql
-- Quel est le taux d'occupation actuel des logements ?
SELECT
    (COUNT(DISTINCT r.id_logement)::FLOAT / COUNT(l.id_logement)) * 100 AS taux_occupation
FROM
    Logement l
LEFT JOIN
    Reservation r ON r.id_logement = l.id_logement
    AND r.date_fin >= CURRENT_DATE;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 | Page No: 1 of 1

	taux_occupation double precision
1	23.076923076923077

	profession character varying (100)	secteur_activite character varying (100)	nb_prolongations bigint
1	Ingénieur	Informatique	3
2	Médecin	Santé	2
3	Etudiant	Éducation	1

6-Cette requête génère les réservations sur différentes périodes

```sql
-- Quelles sont les tendances de réservation sur différentes périodes (mois, trimestre, année) ?
SELECT
    DATE_TRUNC('month', date_debut) AS mois,
    COUNT(*) AS nombre_reservations
FROM Reservation
GROUP BY mois
ORDER BY mois;

-- autre tendance par trimestre et par anne
SELECT
    DATE_TRUNC('quarter', date_debut) AS trimestre,
    COUNT(*) AS nombre_reservations
```

Data Output | Messages | Notifications

Showing rows: 1 to 10 | Page No: 1 of 1

	mois timestamp with time zone	nombre_reservations bigint
1	2024-01-01 00:00:00+01	2
2	2024-02-01 00:00:00+01	4
3	2024-03-01 00:00:00+01	3
4	2024-04-01 00:00:00+02	1