```
-------------------------------------------
-- Création des tables
-------------------------------------------
-- Suppression des tables si elles existent déjà (ordre inverse des dépendances)
DROP TABLE IF EXISTS EST_MAINTENU CASCADE;
DROP TABLE IF EXISTS PARTICIPE CASCADE;
DROP TABLE IF EXISTS RESERVATION CASCADE;
DROP TABLE IF EXISTS EQUIPEMENT CASCADE;
DROP TABLE IF EXISTS MAINTENANCE CASCADE;
DROP TABLE IF EXISTS EVENEMENT CASCADE;
DROP TABLE IF EXISTS RESIDENT CASCADE;
DROP TABLE IF EXISTS LOGEMENT CASCADE;
DROP TABLE IF EXISTS COMPLEXE CASCADE;

-- Création de la table COMPLEXE
CREATE TABLE COMPLEXE (
  idComplexe SERIAL PRIMARY KEY,
  Ville VARCHAR(100) NOT NULL,
  CodePostal VARCHAR(10) NOT NULL,
  NomDeRue VARCHAR(100),
  NumeroDeRue INT
);

-- Création de la table LOGEMENT
CREATE TABLE LOGEMENT (
  idLogement SERIAL PRIMARY KEY,
  idComplexe INT NOT NULL,
  type VARCHAR(50) NOT NULL,
```

```
  etat VARCHAR(10) DEFAULT 'Disponible' CHECK (etat IN ('Disponible', 'Occupée')),
  CONSTRAINT fk_complexe FOREIGN KEY (idComplexe) REFERENCES
COMPLEXE(idComplexe) ON DELETE CASCADE
);

-- Création de la table EQUIPEMENT
CREATE TABLE EQUIPEMENT (
  idEquipement SERIAL PRIMARY KEY,
  nom VARCHAR(100) NOT NULL,
  etat VARCHAR(10) DEFAULT 'Bon' CHECK (etat IN ('Bon', 'Mauvais')),
  tarifEquipement NUMERIC(10, 2) CHECK (tarifEquipement >= 0),
  idLogement INT NOT NULL,
  CONSTRAINT fk_logement FOREIGN KEY (idLogement) REFERENCES
LOGEMENT(idLogement) ON DELETE CASCADE
);

-- Création de la table RESIDENT
CREATE TABLE RESIDENT (
  idResident SERIAL PRIMARY KEY,
  Nom VARCHAR(50) NOT NULL,
  Prenom VARCHAR(50) NOT NULL,
  NumeroDeTelephone VARCHAR(15) UNIQUE,
  Email VARCHAR(100) UNIQUE NOT NULL
);

-- Création de la table RESERVATION
CREATE TABLE RESERVATION (
  idReservation SERIAL PRIMARY KEY,
  dateEntree DATE NOT NULL,
```

```
  dateSortie DATE NOT NULL,
  idResident INT NOT NULL,
  idLogement INT NOT NULL,
  CONSTRAINT fk_resident FOREIGN KEY (idResident) REFERENCES RESIDENT(idResident) ON DELETE CASCADE,
  CONSTRAINT fk_logement_reservation FOREIGN KEY (idLogement) REFERENCES LOGEMENT(idLogement) ON DELETE CASCADE,
  CONSTRAINT check_dates CHECK (dateSortie > dateEntree) -- Empêcher une date de sortie avant l'entrée
);

-- Création de la table MAINTENANCE
CREATE TABLE MAINTENANCE (
  idMaintenance SERIAL PRIMARY KEY,
  type VARCHAR(50) NOT NULL,
  urgence BOOLEAN DEFAULT FALSE,
  tarifM NUMERIC(10, 2) CHECK (tarifM >= 0)
);

-- Création de la table EVENEMENT
CREATE TABLE EVENEMENT (
  idEvenement SERIAL PRIMARY KEY,
  nom VARCHAR(100) NOT NULL,
  dateEvenement DATE NOT NULL CHECK (dateEvenement >= CURRENT_DATE),  -- Empêche d'ajouter des événements passés
  invite VARCHAR(100),
  lieu VARCHAR(100) NOT NULL
);
```

```
-- Création de la table PARTICIPE (relation entre RESIDENT et EVENEMENT)
CREATE TABLE PARTICIPE (
  idResident INT NOT NULL,
  idEvenement INT NOT NULL,
  PRIMARY KEY (idResident, idEvenement),
  CONSTRAINT fk_resident_participe FOREIGN KEY (idResident) REFERENCES RESIDENT(idResident) ON DELETE CASCADE,
  CONSTRAINT fk_evenement_participe FOREIGN KEY (idEvenement) REFERENCES EVENEMENT(idEvenement) ON DELETE CASCADE
);

-- Création de la table EST_MAINTENU (relation entre EQUIPEMENT et MAINTENANCE)
CREATE TABLE EST_MAINTENU (
  idEquipement INT NOT NULL,
  idMaintenance INT NOT NULL,
  date DATE NOT NULL,
  PRIMARY KEY (idEquipement, idMaintenance),
  CONSTRAINT fk_equipement FOREIGN KEY (idEquipement) REFERENCES EQUIPEMENT(idEquipement) ON DELETE CASCADE,
  CONSTRAINT fk_maintenance FOREIGN KEY (idMaintenance) REFERENCES MAINTENANCE(idMaintenance) ON DELETE CASCADE
);

-------------------------------------------
-- Réponses aux questions
-------------------------------------------
--1. Quels logements sont disponibles pour une période donnée, selon des critères spécifiques (type, emplacement, prix) ?
--On peut créer une vue
-- Notion Principale : VUE
```

```
CREATE VIEW LesLogementDisponible AS
SELECT L.idLogement, L.type, L.etat, L.idComplexe, C.Ville,L.prix
FROM LOGEMENT L
JOIN COMPLEXE C ON L.idComplexe = C.idComplexe
LEFT JOIN RESERVATION R ON L.idLogement = R.idLogement
  AND ('2025-02-01' BETWEEN R.dateEntree AND R.dateSortie
      OR '2025-03-01' BETWEEN R.dateEntree AND R.dateSortie)
WHERE R.idReservation IS NULL;
-- Pour simplifier les futures requêtes et éviter de répéter du code.
SELECT * FROM LesLogementDisponible WHERE type = 'Studio' AND Ville = 'Paris' AND
prix < 1000;


--2. Comment gérer les réservations et attribuer les logements aux nouveaux résidents
en optimisant l’occupation ?
  -- Notions Principales : Triggers et un contrôle d'accés
  --> Vérifier la disponibilité du logement avant une réservation afin d'empêcher une
réservation sur un logement non disponible :
CREATE OR REPLACE FUNCTION verifier_disponibilite_logement()
RETURNS TRIGGER AS $$
BEGIN
   --> Vérifier si le logement est déjà réservé pendant la période
   IF EXISTS (
      SELECT 1
      FROM RESERVATION
      WHERE idLogement = NEW.idLogement
      AND (
         (NEW.dateEntree <= dateSortie AND NEW.dateSortie >= dateEntree)
      )
   ) THEN
```

```
      RAISE EXCEPTION 'Le logement est déjà réservé pendant cette période';
   END IF;

   --> Si le logement est disponible, on continue
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

  --> Trigger qui appelle la fonction avant chaque insertion dans RESERVATION
CREATE TRIGGER trigger_verifier_disponibilite
BEFORE INSERT ON RESERVATION
FOR EACH ROW
EXECUTE FUNCTION verifier_disponibilite_logement();

  --> Vérifier et mettre à jour l'état d'un logement après la fin d'une réservation
CREATE OR REPLACE FUNCTION verifier_etat_logement_fin_reservation()
RETURNS TRIGGER AS $$
BEGIN
  --> Vérifier si le logement est "Occupée" et doit être libéré après 13:00 le jour de sortie
  IF EXISTS (
   SELECT 1
   FROM LOGEMENT l
   JOIN RESERVATION r ON l.idLogement = r.idLogement
   WHERE l.idLogement = NEW.idLogement
   AND l.etat = 'Occupée' AND r.dateSortie = CURRENT_DATE AND LOCALTIME >= TIME
'13:00:00'
  ) THEN
   RAISE EXCEPTION 'Le logement %s doit être libéré.', NEW.idLogement;
```

```
 END IF;

 --> Changer l'état du logement à "Disponible" après la fin de la réservation
 UPDATE LOGEMENT SET etat = 'Disponible' WHERE idLogement = NEW.idLogement;

 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

 --> Déclencheur pour mettre à jour l'état du logement après une réservation
CREATE TRIGGER trigger_verifier_etat_logement_fin_reservation
BEFORE INSERT OR UPDATE ON RESERVATION -- Avant insertion d'une nouvelle réservation ou une modification
FOR EACH ROW
EXECUTE FUNCTION verifier_etat_logement_fin_reservation();

 --> Afficher les logements disponibles et leurs caractéristiques
SELECT l.idLogement, l.type, l.prix, l.idComplexe
FROM LOGEMENT l
WHERE l.etat = 'Disponible';

 --> controle d'accés
CREATE ROLE Admin;
CREATE ROLE Gestionnaire;
CREATE ROLE Client;

GRANT ALL PRIVILEGES ON LOGEMENT TO Admin;
GRANT INSERT, UPDATE, DELETE ON RESERVATION TO Gestionnaire;
```

```
GRANT SELECT ON LOGEMENT, RESERVATION TO Client;


--3. Quels résidents partagent actuellement un logement et quelles sont leurs interactions (participation à des événements, conflits signalés) ?
 -- Notions Principales : Jointures multiple de tables
 --3.1 Identifier les résidents qui partagent actuellement un même logement(PAS DE SORTIE)
SELECT R1.idResident AS Resident1, R2.idResident AS Resident2, L.idLogement
FROM RESERVATION R1
JOIN RESERVATION R2 ON R1.idLogement = R2.idLogement AND R1.idResident <> R2.idResident
JOIN LOGEMENT L ON R1.idLogement = L.idLogement
WHERE CURRENT_DATE BETWEEN R1.dateEntree AND R1.dateSortie
 AND CURRENT_DATE BETWEEN R2.dateEntree AND R2.dateSortie
ORDER BY L.idLogement;

 --3.2 Vérifier la participation des résidents à des événements
SELECT DISTINCT R.idResident, E.nom AS NomEvenement, E.date, L.idLogement
FROM PARTICIPE P
JOIN RESIDENT R ON P.idResident = R.idResident
JOIN EVENEMENT E ON P.idEvenement = E.idEvenement
JOIN RESERVATION Res ON R.idResident = Res.idResident
JOIN LOGEMENT L ON Res.idLogement = L.idLogement
WHERE CURRENT_DATE BETWEEN Res.dateEntree AND Res.dateSortie
ORDER BY L.idLogement, E.date;


--4. Quels logements nécessitent le plus d'interventions de maintenance et pourquoi ?
 -- Notions Principales : Common table expresssion (CTE) et requête imbiquée
```

--> Regrouper les maintenances par logement et type via une CTE(Common Table Expression)

```
WITH MaintenanceCounts AS (
 SELECT l.idLogement, m.type, COUNT(*) AS nombreMaintenance
 FROM LOGEMENT l
 JOIN EQUIPEMENT e ON l.idLogement = e.idLogement
 JOIN EST_MAINTENU EstM ON e.idEquipement = EstM.idEquipement
 JOIN MAINTENANCE m ON m.idMaintenace = EstM.idMaintenace
 GROUP BY l.idLogement, m.type
)
```

--> Sélectionner les logements ayant le nombre maximal de maintenances avec leur type

```
SELECT mc.idLogement, mc.type, mc.nombreMaintenance
FROM MaintenanceCounts mc
WHERE mc.nombreMaintenance = (SELECT MAX(nombreMaintenance) FROM MaintenanceCounts);
```

--5. Quels résidents ont prolongé leur séjour, et comment cela impacte les réservations futures ?

-- Notions Principales : Jointures simples

--5.1 : Identifier les résidents qui ont prolongé leur séjour(PAS DE SORTIE)

```
SELECT r1.idResident, r1.dateEntree, r1.dateSortie, r2.dateEntree AS nouvelle_dateEntree, r2.dateSortie AS nouvelle_dateSortie
FROM RESERVATION r1
JOIN RESERVATION r2 ON r1.idResident = r2.idResident
WHERE r1.dateSortie < r2.dateEntree  -- Vérifier si la première réservation se termine avant le début de la nouvelle réservation
AND r2.dateEntree > r1.dateEntree  -- Assurer qu'il y a prolongation (pas de nouvelle réservation en dehors de la période initiale)
```

AND r1.dateSortie < CURRENT_DATE; -- Les prolongations qui concernent des réservations déjà passées

--5.2 : Analyser l'impact des prolongations sur les réservations futures(PAS DE SORTIE)

SELECT r1.idResident, l.idLogement, r1.dateEntree, r1.dateSortie, r2.dateEntree AS nouvelle_dateEntree, r2.dateSortie AS nouvelle_dateSortie

FROM RESERVATION r1

JOIN RESERVATION r2 ON r1.idResSELECT DISTINCT R1.idResident AS Resident1, R2.idResident AS Resident2, L.idLogement, E.nom AS NomEvenement, E.date

ident = r2.idResident

JOIN LOGEMENT l ON r1.idLogement = l.idLogement

WHERE r1.dateSortie < r2.dateEntree

AND r2.dateEntree > r1.dateEntree

AND r1.dateSortie < CURRENT_DATE

AND r2.dateEntree > CURRENT_DATE; -- Vérification l'impact des prolongations sur les réservations futures

--6. Comment organiser les événements communautaires pour maximiser la participation des résidents dans un logement donné ?

-- Notions Principales : Jointure et Trigger

-->Evenements populaires :

SELECT e.nom AS evenement, COUNT(p.idResident) AS nombre_participants

FROM EVENEMENT e

LEFT JOIN PARTICIPE p ON e.idEvenement = p.idEvenement

GROUP BY e.idEvenement

ORDER BY nombre_participants DESC;

--> Les résidents qui ne participent pas aux événements :

```
SELECT r.Nom, r.Prenom, r.Email
FROM RESIDENT r
LEFT JOIN PARTICIPE p ON r.idResident = p.idResident  -- Tous les résidents, indépendamment de leur participation
WHERE p.idResident IS NULL; -- Ceux qui ne participent pas


 --> Informer sur les prochains événements :
CREATE OR REPLACE FUNCTION notifier_nouvel_evenement()
RETURNS TRIGGER AS $$
BEGIN
 -- Déclencher une nouvelle notice pour informer
  RAISE NOTICE 'Nouvel événement : % prévu le % !', NEW.nom, NEW.dateEvenement;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_notifier_evenement
AFTER INSERT ON EVENEMENT -- Après ajout d'un nouvel événement
FOR EACH ROW
EXECUTE FUNCTION notifier_nouvel_evenement();


--7. Quels types de logements sont les plus demandés et quelles améliorations peuvent augmenter leur attractivité ?
 -- Notions Principales : Utilisation d'aggrégats
 --7. 1 : Identifier les types de logements les plus demandés
SELECT l.type, COUNT(r.idReservation) AS nombreReservations
FROM LOGEMENT l
JOIN RESERVATION r ON l.idLogement = r.idLogement
```

```
GROUP BY l.type
ORDER BY nombreReservations DESC;

 --7.2 : Analyser les améliorations possibles pour augmenter l'attractivité des logements(ATTENTION:PAS DE SORTIE)
SELECT l.type, e.idEquipement, COUNT(r.idReservation) AS nombreReservations
FROM LOGEMENT l
JOIN RESERVATION r ON l.idLogement = r.idLogement
JOIN EQUIPEMENT e  ON l.idLogement = e.idLogement
GROUP BY l.type, e.idEquipement
ORDER BY nombreReservations DESC;

 --7.3 : Suggestions d'améliorations pour augmenter l'attractivité
SELECT l.type, COUNT(r.idReservation) AS nombreReservations,
    AVG(e.tarifEquipement) AS moyennePrix,
    MAX(e.etat) AS meilleureQualite
FROM LOGEMENT l
JOIN RESERVATION r ON l.idLogement = r.idLogement
JOIN EQUIPEMENT e ON e.idEquipement = e.idEquipement
GROUP BY l.type
ORDER BY nombreReservations DESC;
-------------------------------------------
-- Insertion de données général
-------------------------------------------
-- Insertion des 8 complexes
INSERT INTO COMPLEXE (Ville, CodePostal, NomDeRue, NumeroDeRue) VALUES
('Paris', '75001', 'Rue de Rivoli', 10),
('Lyon', '69001', 'Rue de la République', 20),
```

('Marseille', '13001', 'Cours Belsunce', 30),

('Bordeaux', '33000', 'Place des Quinconces', 40),

('Toulouse', '31000', 'Avenue de la République', 50),

('Nice', '06000', 'Promenade des Anglais', 60),

('Nantes', '44000', 'Boulevard de la Liberté', 70),

('Strasbourg', '67000', 'Rue du Vieux Marché aux Poissons', 80);

-- Insertion des 160 logements (20 par complexe)

INSERT INTO LOGEMENT (idComplexe, type, etat, prix) VALUES

-- Complexe 1

(1, 'Studio', 'Disponible', 800),

(1, 'Studio', 'Disponible', 820),

(1, 'Appartement T1', 'Disponible', 950),

(1, 'Appartement T1', 'Disponible', 970),

(1, 'Appartement T2', 'Occupée', 1200),

(1, 'Appartement T2', 'Disponible', 1250),

(1, 'Appartement T3', 'Disponible', 1500),

(1, 'Appartement T3', 'Disponible', 1550),

(1, 'Loft', 'Occupée', 2000),

(1, 'Loft', 'Disponible', 2100),

(1, 'Villa', 'Disponible', 3500),

(1, 'Villa', 'Disponible', 3600),

(1, 'Penthouse', 'Occupée', 5000),

(1, 'Penthouse', 'Disponible', 5200),

(1, 'Duplex', 'Disponible', 2800),

(1, 'Duplex', 'Disponible', 2900),

(1, 'Maison de ville', 'Disponible', 3200),

(1, 'Maison de ville', 'Disponible', 3300),

(1, 'Chambre individuelle', 'Disponible', 500),

(1, 'Chambre individuelle', 'Disponible', 520),

-- Complexe 2

(2, 'Studio', 'Disponible', 750),

(2, 'Studio', 'Disponible', 770),

(2, 'Appartement T1', 'Occupée', 920),

(2, 'Appartement T1', 'Disponible', 940),

(2, 'Appartement T2', 'Disponible', 1100),

(2, 'Appartement T2', 'Disponible', 1150),

(2, 'Appartement T3', 'Disponible', 1400),

(2, 'Appartement T3', 'Occupée', 1450),

(2, 'Loft', 'Disponible', 1900),

(2, 'Loft', 'Disponible', 1950),

(2, 'Villa', 'Disponible', 3400),

(2, 'Villa', 'Occupée', 3500),

(2, 'Penthouse', 'Disponible', 4900),

(2, 'Penthouse', 'Disponible', 5000),

(2, 'Duplex', 'Disponible', 2700),

(2, 'Duplex', 'Occupée', 2750),

(2, 'Maison de ville', 'Disponible', 3100),

(2, 'Maison de ville', 'Disponible', 3150),

(2, 'Chambre individuelle', 'Disponible', 480),

(2, 'Chambre individuelle', 'Disponible', 490),

-- Complexe 3

(3, 'Studio', 'Disponible', 780),

(3, 'Studio', 'Disponible', 800),

(3, 'Appartement T1', 'Disponible', 970),

(3, 'Appartement T1', 'Disponible', 990),

(3, 'Appartement T2', 'Occupée', 1250),

(3, 'Appartement T2', 'Disponible', 1300),

(3, 'Appartement T3', 'Disponible', 1600),

(3, 'Appartement T3', 'Disponible', 1650),

(3, 'Loft', 'Occupée', 2100),

(3, 'Loft', 'Disponible', 2200),

(3, 'Villa', 'Disponible', 3600),

(3, 'Villa', 'Disponible', 3700),

(3, 'Penthouse', 'Occupée', 5300),

(3, 'Penthouse', 'Disponible', 5500),

(3, 'Duplex', 'Disponible', 3000),

(3, 'Duplex', 'Disponible', 3100),

(3, 'Maison de ville', 'Disponible', 3400),

(3, 'Maison de ville', 'Disponible', 3500),

(3, 'Chambre individuelle', 'Disponible', 530),

(3, 'Chambre individuelle', 'Disponible', 550),

-- Complexe 4

(4, 'Studio', 'Disponible', 790),

(4, 'Studio', 'Disponible', 810),

(4, 'Appartement T1', 'Occupée', 980),

(4, 'Appartement T1', 'Disponible', 1000),

(4, 'Appartement T2', 'Disponible', 1270),

(4, 'Appartement T2', 'Disponible', 1320),

(4, 'Appartement T3', 'Disponible', 1650),

(4, 'Appartement T3', 'Occupée', 1700),

(4, 'Loft', 'Disponible', 2150),

(4, 'Loft', 'Disponible', 2250),

(4, 'Villa', 'Disponible', 3650),

(4, 'Villa', 'Occupée', 3750),

(4, 'Penthouse', 'Disponible', 5400),

(4, 'Penthouse', 'Disponible', 5600),

(4, 'Duplex', 'Disponible', 3100),

(4, 'Duplex', 'Occupée', 3200),

(4, 'Maison de ville', 'Disponible', 3500),

(4, 'Maison de ville', 'Disponible', 3600),

(4, 'Chambre individuelle', 'Disponible', 550),

(4, 'Chambre individuelle', 'Disponible', 570),

-- Complexe 5

(5, 'Studio', 'Disponible', 800),

(5, 'Studio', 'Disponible', 820),

(5, 'Appartement T1', 'Disponible', 990),

(5, 'Appartement T1', 'Disponible', 1010),

(5, 'Appartement T2', 'Occupée', 1290),

(5, 'Appartement T2', 'Disponible', 1340),

(5, 'Appartement T3', 'Disponible', 1700),

(5, 'Appartement T3', 'Disponible', 1750),

(5, 'Loft', 'Occupée', 2200),

(5, 'Loft', 'Disponible', 2300),

(5, 'Villa', 'Disponible', 3700),

(5, 'Villa', 'Disponible', 3800),

(5, 'Penthouse', 'Occupée', 5500),

(5, 'Penthouse', 'Disponible', 5700),

(5, 'Duplex', 'Disponible', 3200),

(5, 'Duplex', 'Disponible', 3300),

(5, 'Maison de ville', 'Disponible', 3600),

(5, 'Maison de ville', 'Disponible', 3700),

(5, 'Chambre individuelle', 'Disponible', 570),

(5, 'Chambre individuelle', 'Disponible', 590),

-- Complexe 6

(6, 'Studio', 'Disponible', 810),

(6, 'Studio', 'Disponible', 830),

(6, 'Appartement T1', 'Occupée', 1000),

(6, 'Appartement T1', 'Disponible', 1020),

(6, 'Appartement T2', 'Disponible', 1310),

(6, 'Appartement T2', 'Disponible', 1360),

(6, 'Appartement T3', 'Disponible', 1750),

(6, 'Appartement T3', 'Occupée', 1800),

(6, 'Loft', 'Disponible', 2250),

(6, 'Loft', 'Disponible', 2350),

(6, 'Villa', 'Disponible', 3750),

(6, 'Villa', 'Occupée', 3850),

(6, 'Penthouse', 'Disponible', 5600),

(6, 'Penthouse', 'Disponible', 5800),

(6, 'Duplex', 'Disponible', 3300),

(6, 'Duplex', 'Occupée', 3400),

(6, 'Maison de ville', 'Disponible', 3700),

(6, 'Maison de ville', 'Disponible', 3800),

(6, 'Chambre individuelle', 'Disponible', 590),

(6, 'Chambre individuelle', 'Disponible', 610),

-- Complexe 7

(7, 'Studio', 'Disponible', 820),

(7, 'Studio', 'Disponible', 840),

(7, 'Appartement T1', 'Disponible', 1010),

(7, 'Appartement T1', 'Disponible', 1030),

(7, 'Appartement T2', 'Occupée', 1330),

(7, 'Appartement T2', 'Disponible', 1380),

(7, 'Appartement T3', 'Disponible', 1800),

(7, 'Appartement T3', 'Disponible', 1850),

(7, 'Loft', 'Occupée', 2300),

(7, 'Loft', 'Disponible', 2400),

(7, 'Villa', 'Disponible', 3800),

(7, 'Villa', 'Disponible', 3900),

(7, 'Penthouse', 'Occupée', 5700),

(7, 'Penthouse', 'Disponible', 5900),

(7, 'Duplex', 'Disponible', 3400),

(7, 'Duplex', 'Disponible', 3500),

(7, 'Maison de ville', 'Disponible', 3800),

(7, 'Maison de ville', 'Disponible', 3900),

(7, 'Chambre individuelle', 'Disponible', 610),

(7, 'Chambre individuelle', 'Disponible', 630),

-- Complexe 8

(8, 'Studio', 'Disponible', 830),

(8, 'Studio', 'Disponible', 850),

(8, 'Appartement T1', 'Occupée', 1020),

(8, 'Appartement T1', 'Disponible', 1040),

(8, 'Appartement T2', 'Disponible', 1350),

(8, 'Appartement T2', 'Disponible', 1400),

(8, 'Appartement T3', 'Disponible', 1850),

(8, 'Appartement T3', 'Occupée', 1900),

(8, 'Loft', 'Disponible', 2350),

(8, 'Loft', 'Disponible', 2450),

(8, 'Villa', 'Disponible', 3850),

(8, 'Villa', 'Occupée', 3950),

(8, 'Penthouse', 'Disponible', 5800),

(8, 'Penthouse', 'Disponible', 6000),

(8, 'Duplex', 'Disponible', 3500),

(8, 'Duplex', 'Occupée', 3600),

(8, 'Maison de ville', 'Disponible', 3900),

(8, 'Maison de ville', 'Disponible', 4000),

(8, 'Chambre individuelle', 'Disponible', 630),

(8, 'Chambre individuelle', 'Disponible', 650);

-- Insertion des résidents

INSERT INTO RESIDENT (Nom, Prenom, NumeroDeTelephone, Email) VALUES

('Dupont', 'Jean', '0123456789', 'jean.dupont@example.com'),

('Martin', 'Marie', '0234567890', 'marie.martin@example.com'),

('Durand', 'Pierre', '0345678901', 'pierre.durand@example.com'),

('Lemoine', 'Paul', '0456789012', 'paul.lemoine@example.com'),

('Robert', 'Sophie', '0567890123', 'sophie.robert@example.com'),

('Dubois', 'Lucie', '0678901234', 'lucie.dubois@example.com'),

('Garnier', 'Julien', '0789012345', 'julien.garnier@example.com'),

('Lemoine', 'Claire', '0890123456', 'claire.lemoine@example.com'),

('Dufresne', 'Nathalie', '0901234567', 'nathalie.dufresne@example.com'),

('Moreau', 'Thomas', '0912345678', 'thomas.moreau@example.com'),

('Bernard', 'Camille', '0923456789', 'camille.bernard@example.com'),

('Leroy', 'Noah', '0934567890', 'noah.leroy@example.com'),

('Guerin', 'Lucas', '0945678901', 'lucas.guerin@example.com'),

('Blanc', 'Hugo', '0956789012', 'hugo.blanc@example.com'),

('Meyer', 'Chloé', '0967890123', 'chloe.meyer@example.com'),

('Carpentier', 'Léa', '0978901234', 'lea.carpentier@example.com'),

('Garcia', 'Mathieu', '0989012345', 'mathieu.garcia@example.com'),

('Robin', 'Elodie', '0990123456', 'elodie.robin@example.com'),

('Henry', 'Victor', '0991234567', 'victor.henry@example.com'),

('Barbier', 'Sarah', '0992345678', 'sarah.barbier@example.com'),

('Lambert', 'Emma', '0993456789', 'emma.lambert@example.com');

-- Insertion des réservations

INSERT INTO RESERVATION (dateEntree, dateSortie, idResident, idLogement) VALUES

('2025-02-01', '2025-02-28', 1, 153),

('2025-02-01', '2025-02-28', 2, 151),

('2025-02-01', '2025-02-28', 3, 155),

('2025-02-01', '2025-02-28', 4, 157),

('2025-02-01', '2025-02-28', 5, 159),

('2025-02-01', '2025-02-28', 6, 137),

('2025-02-01', '2025-02-28', 7, 133),

('2025-02-01', '2025-02-28', 8, 131),

-- Jean Dupont et Marie Martin partagent le logement 2

('2025-02-01', '2025-02-28', 1, 2),

('2025-02-01', '2025-02-28', 2, 2),

-- Paul Lemoine et Sophie Robert partagent le logement 3

('2025-02-05', '2025-03-05', 3, 3),

('2025-02-10', '2025-03-10', 4, 3),

-- Jean Dupont réserve et prolonge son séjour

('2025-01-01', '2025-01-15', 1, 1),

('2025-01-15', '2025-01-30', 1, 1),

-- Marie Martin réserve **mais elle commence avant que Jean Dupont parte**

('2025-01-28', '2025-02-10', 2, 1), -- CONFLIT : elle arrive avant le départ de Jean Dupont

('2025-02-01', '2025-02-28', 1, 1),

('2025-02-01', '2025-02-28', 2, 1),

('2025-02-01', '2025-02-28', 3, 2),

('2025-02-01', '2025-02-28', 4, 3),

('2025-02-01', '2025-02-28', 5, 4);

-- Insertion des maintenances

INSERT INTO MAINTENANCE (type, urgence, tarifM) VALUES

('Réparation de plomberie', TRUE, 100.00),

('Réparation de chauffage', FALSE, 80.00),

('Réparation de réfrigérateur', TRUE, 120.00),

('Peinture', FALSE, 50.00),

('Réparation électrique', TRUE, 150.00);

-- Insertion des événements

INSERT INTO EVENEMENT (nom, date, invite, lieu) VALUES

('Réunion des résidents', '2025-02-10', 'Tous les résidents', 'Salle commune du complexe 1'),

('Soirée barbecue', '2025-02-15', 'Tous les résidents', 'Jardin du complexe 2'),

('Atelier peinture', '2025-02-20', 'Résidents intéressés', 'Salle des loisirs du complexe 3'),

('Soirée cinéma', '2025-02-25', 'Tous les résidents', 'Salle commune du complexe 4'),

('Chasse au trésor', '2025-02-28', 'Tous les résidents', 'Terrains de sport du complexe 5');

-- Insertion des participations

INSERT INTO PARTICIPE (idResident, idEvenement) VALUES

(1, 1),

(2, 1),

(3, 2),

(4, 2),

(5, 3),

(6, 3),

(7, 4),

(8, 4);

-- Insertion des maintenances sur les équipements

INSERT INTO EST_MAINTENU (idEquipement, idMaintenance, date) VALUES

(3, 3, '2025-02-05'),

(10, 1, '2025-02-06'),

(12, 2, '2025-02-07'),

(17, 5, '2025-02-08');

INSERT INTO EQUIPEMENT (nom, etat, tarifEquipement, idLogement) VALUES

('Climatisation', 'Bon', 200, 1),

('Chauffage', 'Mauvais', 150, 1),

('Réfrigérateur', 'Bon', 300, 2),

('Four', 'Mauvais', 180, 2),

('WiFi', 'Bon', 50, 3),

('Lave-linge', 'Mauvais', 250, 4),

('Climatisation', 'Bon', 200, 1),

('Chauffage', 'Mauvais', 150, 1),

('Réfrigérateur', 'Bon', 300, 2),

('Four', 'Mauvais', 180, 2),

('WiFi', 'Bon', 50, 3),

('Lave-linge', 'Mauvais', 250, 3);

-- Insertion de données pour les réponses aux questions 2, 4 et 6 par Guilaye pour valider les requêtes

---

-- ---

-- Jeu de données pour tester mes requêtes

-- ---

-- Insérer les complexes

INSERT INTO COMPLEXE (Ville, CodePostal, NomDeRue, NumeroDeRue)

```
VALUES
('Paris', '75001', 'Rue de Rivoli', 1),
('Lyon', '69001', 'Avenue des Brotteaux', 2);

-- Insérer les logements associés aux complexes
INSERT INTO LOGEMENT (idComplexe, type, etat)
VALUES
(1, 'Studio', 'Disponible'),
(1, 'T2', 'Occupée'),
(2, 'Studio', 'Disponible'),
(2, 'T3', 'Disponible');

-- Insérer les équipements pour les logements
INSERT INTO EQUIPEMENT (nom, etat, tarifEquipement, idLogement)
VALUES
('Télévision', 'Bon', 15.00, 1),
('Frigo', 'Bon', 20.00, 1),
('Machine à laver', 'Mauvais', 10.00, 2),
('Four', 'Bon', 25.00, 3);

-- Insérer les résidents
INSERT INTO RESIDENT (Nom, Prenom, NumeroDeTelephone, Email)
VALUES
('Dupont', 'Pierre', '0601020304', 'pierre.dupont@email.com'),
('Martin', 'Sophie', '0605060708', 'sophie.martin@email.com'),
('Lemoine', 'Marc', '0612345678', 'marc.lemoine@email.com');

-- Insérer des réservations
```

```
INSERT INTO RESERVATION (dateEntree, dateSortie, idResident, idLogement)
VALUES
('2025-02-05', '2025-02-10', 1, 1),
('2025-02-06', '2025-02-12', 2, 2),
('2025-02-07', '2025-02-15', 3, 3);

-- Insérer des maintenances pour les équipements
INSERT INTO MAINTENANCE (type, urgence, tarifM)
VALUES
('Réparation Frigo', FALSE, 30.00),
('Entretien Machine à laver', TRUE, 40.00),
('Réparation Télévision', FALSE, 20.00);

-- Insérer les associations de maintenance pour les équipements
INSERT INTO EST_MAINTENU (idEquipement, idMaintenance, date)
VALUES
(1, 1, '2025-02-05'),
(2, 2, '2025-02-06'),
(3, 1, '2025-02-07');

-- Insérer des événements
INSERT INTO EVENEMENT (nom, dateEvenement, invite, lieu)
VALUES
('Réunion de Quartier', '2025-02-10', 'Tous les résidents', 'Salle de réunion 1'),
('Soirée Cinéma', '2025-02-15', 'Tous les résidents', 'Salle de cinéma');

-- Insérer des participations aux événements
INSERT INTO PARTICIPE (idResident, idEvenement)
```

```
VALUES
(1, 1), -- Pierre participe à la réunion
(2, 1), -- Sophie participe à la réunion
(3, 2); -- Marc participe à la soirée cinéma

-- ----------------------------------
-- Tests pour chaque question
-- ----------------------------------

-- Question 2 : Tester la vérification de la disponibilité d'un logement avant une réservation
-- Essaie d'insérer une réservation pour un logement déjà occupé (idLogement = 2, déjà occupé)
INSERT INTO RESERVATION (dateEntree, dateSortie, idResident, idLogement)
VALUES ('2025-02-11', '2025-02-14', 1, 2);
-- Cela échouera normalement avec le trigger en place
--------------------------------------------------------------------------------
-- ERROR:  Le logement est déjà réservé pendant cette période
-- CONTEXT:  fonction PL/pgSQL verifier_disponibilite_logement(), ligne 12 à RAISE

-- ERREUR:  Le logement est déjà réservé pendant cette période
--------------------------------------------------------------------------------

-- Question 4 : Tester les maintenances les plus fréquentes
WITH MaintenanceCounts AS (
  SELECT l.idLogement, m.type, COUNT(*) AS nombreMaintenance
  FROM LOGEMENT l
  JOIN EQUIPEMENT e ON l.idLogement = e.idLogement
  JOIN EST_MAINTENU EstM ON e.idEquipement = EstM.idEquipement
```

```
  JOIN MAINTENANCE m ON m.idMaintenance = EstM.idMaintenance
  GROUP BY l.idLogement, m.type
)
SELECT mc.idLogement, mc.type, mc.nombreMaintenance
FROM MaintenanceCounts mc
WHERE mc.nombreMaintenance = (SELECT MAX(nombreMaintenance) FROM
MaintenanceCounts);

-- Résultats attendu :
-----------------------------------------------------------------
--idlogement       "type"            "nombremaintenance"
-----------------------------------------------------------------
-- 1        "Entretien Machine à laver"       "1"
-- 1         "Réparation Frigo"            "1"
-- 2         "Réparation Frigo"            "1"
-----------------------------------------------------------------

-- Question 6 : Tester la participation aux événements
SELECT e.nom AS evenement, COUNT(p.idResident) AS nombre_participants
FROM EVENEMENT e
LEFT JOIN PARTICIPE p ON e.idEvenement = p.idEvenement
GROUP BY e.idEvenement
ORDER BY nombre_participants DESC;

-- Résultats attendu :
-------------------------------------------------
--     evenement       "nombre_participants"
-------------------------------------------------
```

-- Réunion de Quartier "2"

-- Soirée Cinéma "1"

-- Tester les résidents qui ne participent à aucun événement

SELECT r.Nom, r.Prenom, r.Email

FROM RESIDENT r

LEFT JOIN PARTICIPE p ON r.idResident = p.idResident

WHERE p.idResident IS NULL;

-- Normalement chacun à participer à au moins 1 évènement donc le résultat devra être vide au cas où vous lancerez ce test

-- Notifier un nouvel événement : Ajouter un nouvel événement pour tester la notification

INSERT INTO EVENEMENT (nom, dateEvenement, invite, lieu)

VALUES ('Atelier Cuisine', '2025-02-20', 'Tous les résidents', 'Salle de loisirs');

-- Sortie :

-- NOTICE: Nouvel événement : Atelier Cuisine prévu le 2025-02-20 !

-- INSERT 0 1

-- Query returned successfully in 112 msec
