## Exercice. Modèle relationnel

- **Table LIEU**

| Colonne | Type | Contrainte |
|---|---|---|
| **id_lieu** | **INT** | **Identifiant unique du lieu, généré automatiquement** |
| **nom_lieu** | **VARCHAR(100** | **Obligatoire** |
| **adresse** | **VARCHAR(200** | **Obligatoire** |
| **capacite** | **INT** | **Obligatoire** |
| **date_ouverture** | **DATE** | **Obligatoire** |
| **heure_ouverture** | **TIME** | **Optionnel (peut être non renseigné)** |
| **heure_fermeture** | **TIME** | **Optionnel (peut être non renseigné)** |

- **Table CATEGORIE**

| Colonne | Type | Contrainte |
|---|---|---|
| **id_categorie** | **INT** | **Identifiant unique de la catégorie, généré automatiquement** |
| **nom_categorie** | **VARCHAR(50)** | **Obligatoire et doit être unique dans toute la base** |
| **description** | **TEXT** | **Optionnel (peut être non renseigné)** |
| **date_creation** | **DATE** | **Obligatoire** |

- **Table EVENEMENT**

| Colonne | Type | Contrainte en langage naturel |
|---|---|---|
| **id_evenement** | **INT** | **Identifiant unique de l'événement, généré automatiquement** |
| **titre** | **VARCHAR(100)** | **Obligatoire** |
| **id_lieu** | **INT** | **Référence à un lieu existant (clé étrangère)** |
| **id_categorie** | **INT** | **Référence à une catégorie existante (clé étrangère)** |
| **date_debut** | **DATETIME** | **Obligatoire** |
| **date_fin** | **DATETIME** | **Obligatoire** |

| | | |
|---|---|---|
| **prix_base** | **DECIMAL(8,2)** | **Obligatoire** |
| **date_publication** | **DATE** | **Obligatoire** |
| **date_cloture_reservations** | **DATE** | **Obligatoire** |

***Contraintes supplémentaires* :**

- **La date de fin doit être postérieure à la date de début**
- **La date de clôture des réservations doit être antérieure ou égale à la date de début**

### ❑ Table PARTICIPANT

| Colonne | Type | Contrainte en langage naturel |
|---|---|---|
| **id_participant** | **INT** | **Identifiant unique du participant, généré automatiquement** |
| **nom** | **VARCHAR(50)** | **Obligatoire** |
| **prenom** | **VARCHAR(50)** | **Obligatoire** |
| **email** | **VARCHAR(100)** | **Obligatoire et doit être unique dans toute la base** |
| **date_naissance** | **DATE** | **Obligatoire** |
| **telephone** | **VARCHAR(15)** | **Optionnel (peut être non renseigné)** |
| **date_inscription** | **DATE** | **Obligatoire** |
| **derniere_connexion** | **DATE** | **Optionnel (peut être non renseigné)** |

### ❑ Table RESERVATION

| Colonne | Type | Contrainte en langage naturel |
|---|---|---|
| **id_reservation** | **INT** | **Identifiant unique de la réservation, généré automatiquement** |
| **id_evenement** | **INT** | **Référence à un événement existant (clé étrangère)** |
| **id_participant** | **INT** | **Référence à un participant existant (clé étrangère)** |
| **date_reservation** | **DATE** | **Obligatoire** |
| **nb_places** | **INT** | **Obligatoire, valeur par défaut: 1** |
| **statut** | **Varchar(20)** | **Obligatoire, doit être l'une des valeurs suivantes: 'confirmée', 'annulée', 'en attente'.** |

| | | |
|---|---|---|
| **date_paiement** | **DATE** | **Optionnel (peut être non renseigné)** |
| **date_annulation** | **DATE** | **Optionnel (peut être non renseigné)** |

**Questions : Donner les requêtes SQL qui permet de :**

1- Définir et créer les tables suivantes en respectant les contraintes associées.
2- Créer une vue affichant les événements avec leur taux d'occupation, en incluant le nombre total de réservations, la capacité du lieu et une classification du taux d'occupation tel que les 3 catégories du taux d'occupation sont :
   - **Faible** : Moins de 50% de la capacité.
   - **Moyen** : Entre 50% et 75% de la capacité.
   - **Élevé** : Plus de 75% de la capacité.
3- Afficher la liste des événements avec leur lieu et catégorie
4- Rechercher les événements à venir
5- Identifier les participants inactifs depuis plus de 6 mois.
6- Mettre à jour le statut d'une réservation en modifiant le statut d'une réservation à « confirmée » et enregistrer la date de paiement.
7- Affichez pour chaque participant le nombre d'événements pour lesquels il a une réservation.
8- Afficher les **participants** ayant effectué une **réservation** après le **1er janvier 2023** pour des **événements** dont la **date de début** est également après cette même date.
9- Supprimer les réservations qui ont été effectuées il y a plus de 6 mois.

## La correction

1- Définir et créer les tables en respectant les contraintes associées

```
CREATE TABLE LIEU (
    id_lieu INT PRIMARY KEY AUTO_INCREMENT,
    nom_lieu VARCHAR(100) NOT NULL,
    adresse VARCHAR(200) NOT NULL,
    capacite INT NOT NULL,
    date_ouverture DATE NOT NULL,
    heure_ouverture TIME,
    heure_fermeture TIME
);

CREATE TABLE CATEGORIE (
    id_categorie INT PRIMARY KEY AUTO_INCREMENT,
    nom_categorie VARCHAR(50) NOT NULL UNIQUE,
    description TEXT,
    date_creation DATE NOT NULL
);

CREATE TABLE EVENEMENT (
    id_evenement INT PRIMARY KEY AUTO_INCREMENT,
    titre VARCHAR(100) NOT NULL,
    id_lieu INT,
    id_categorie INT,
    date_debut DATETIME NOT NULL,
    date_fin DATETIME NOT NULL,
    prix_base DECIMAL(8,2) NOT NULL,
    date_publication DATE NOT NULL,
    date_cloture_reservations DATE NOT NULL,
    FOREIGN KEY (id_lieu) REFERENCES LIEU(id_lieu),
    FOREIGN KEY (id_categorie) REFERENCES CATEGORIE(id_categorie),
    CONSTRAINT chk_date_fin CHECK (date_fin > date_debut),
    CONSTRAINT chk_date_cloture CHECK (date_cloture_reservations <=
date_debut)
);

CREATE TABLE PARTICIPANT (
    id_participant INT PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(50) NOT NULL,
    prenom VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    date_naissance DATE NOT NULL,
    telephone VARCHAR(15),
    date_inscription DATE NOT NULL,
    derniere_connexion DATE
);
```

```
CREATE TABLE RESERVATION (
    id_reservation INT PRIMARY KEY AUTO_INCREMENT,
    id_evenement INT,
    id_participant INT,
    date_reservation DATE NOT NULL,
    nb_places INT NOT NULL DEFAULT 1,
    statut VARCHAR(20) NOT NULL CHECK (statut IN ('confirmée', 'annulée',
'en attente')),
    date_paiement DATE,
    date_annulation DATE,
    FOREIGN KEY (id_evenement) REFERENCES EVENEMENT(id_evenement),
    FOREIGN KEY (id_participant) REFERENCES PARTICIPANT(id_participant)
);
```

2. Créer une vue affichant les événements avec leur taux d'occupation

```
CREATE VIEW Vue_Taux_Occupation AS
SELECT
    e.id_evenement,
    e.titre,
    l.capacite,
    SUM(r.nb_places) AS total_reservations,
    CASE
        WHEN SUM(r.nb_places)/ l.capacite < 0.5 * l.capacite THEN 'Faible'
        WHEN SUM(r.nb_places) )/ l.capacite BETWEEN 0.5 * l.capacite AND
0.75 * l.capacite THEN 'Moyen'
        ELSE 'Élevé'
    END AS taux_occupation
FROM EVENEMENT e
JOIN LIEU l ON e.id_lieu = l.id_lieu
JOIN RESERVATION r ON e.id_evenement = r.id_evenement
GROUP BY e.id_evenement, l.capacite;
```

**3-** Afficher la liste des événements avec leur lieu et catégorie

```
SELECT
    e.titre AS evenement,
    l.nom_lieu AS lieu,
    c.nom_categorie AS categorie
FROM EVENEMENT e
JOIN LIEU l ON e.id_lieu = l.id_lieu
JOIN CATEGORIE c ON e.id_categorie = c.id_categorie;
```

**4- Rechercher les événements à venir**

```
SELECT
    titre,
```

```
    date_debut,
    date_fin
FROM EVENEMENT
WHERE date_debut > SYSDATE;
```

**5- Identifier les participants inactifs depuis plus de 6 mois**

```
SELECT
    id_participant,
    nom,
    prenom,
    derniere_connexion
FROM PARTICIPANT
WHERE MONTHS_BETWEEN(SYSDATE, derniere_connexion) > 6
OR derniere_connexion IS NULL;
```

**6- Mettre à jour le statut d'une réservation d'ID 18**

```
UPDATE RESERVATION
SET statut = 'confirmée',
    date_paiement = SYSDATE
WHERE id_reservation = 18;
```

**7- Afficher pour chaque participant le nombre d'événements réservés**

```
SELECT
    p.id_participant,
    p.nom,
    p.prenom,
    COUNT(r.id_evenement) AS nombre_evenements_reserves
FROM PARTICIPANT p
LEFT JOIN RESERVATION r ON p.id_participant = r.id_participant
GROUP BY p.id_participant;
```

**8- Afficher les participants ayant effectué une réservation après le 1er janvier 2023 pour des événements dont la date de début est également après cette même date.**

```
SELECT DISTINCT
    p.id_participant,
    p.nom,
    p.prenom
FROM PARTICIPANT p
JOIN RESERVATION r ON p.id_participant = r.id_participant
JOIN EVENEMENT e ON r.id_evenement = e.id_evenement
WHERE r.date_reservation > '2023-01-01'
AND e.date_debut > '2023-01-01';
```

**9- Supprimer les réservations qui ont été effectuées il y a plus de 6 mois.**

```
DELETE FROM RESERVATION
WHERE MONTHS_BETWEEN(SYSDATE, date_reservation) > 6;
```

10. le nombre total de réservations par événement et ajouter cette information à chaque ligne de réservation

```
SELECT
    r.id_reservation,
    r.id_evenement,
    r.id_participant,
    r.date_reservation,
    r.nb_places,
    r.statut,
    COUNT(*)      OVER      (PARTITION      BY      r.id_evenement)      AS
total_reservations_par_evenement
FROM RESERVATION r;
```

11. Classer les événements par prix dans chaque catégorie

```
SELECT
    e.id_evenement,
    e.titre,
    e.id_categorie,
    e.prix_base,
    DENSE_RANK() OVER (PARTITION BY e.id_categorie ORDER BY e.prix_base
DESC) AS classement_prix
FROM EVENEMENT e;
```