Compte Rendu Projet PL/SQL

Projet réalisé par AMOUSSOU Nathan, LAM Mathieu, JEYAKANTHAN Thushanth.

A. Création du schéma de la base de données

```
CREATE TABLE SalleMusculation (
    ID_Salle NUMBER NOT NULL,
    Nom VARCHAR2(100) NOT NULL,
    Adresse VARCHAR2(255) NOT NULL UNIQUE,
    DateOuverture DATE NOT NULL,
    SurfaceTotale NUMBER NOT NULL CHECK (SurfaceTotale > 0),
    HoraireOuverture TIMESTAMP NOT NULL,
    HoraireFermeture TIMESTAMP NOT NULL,
    PRIMARY KEY (ID_Salle),
    CHECK (HoraireOuverture < HoraireFermeture)
);

CREATE TABLE Abonnement (
    ID_Abonnement NUMBER NOT NULL,
    Type_Abonnement VARCHAR2(50) NOT NULL,
    Prix NUMBER NOT NULL CHECK (Prix >= 0),
    DateDebut DATE NOT NULL,
    DateFin DATE,
    PRIMARY KEY (ID_Abonnement)
);

CREATE TABLE Membre (
    ID_Membre NUMBER NOT NULL,
    ID_Abonnement NUMBER NOT NULL,
    Nom VARCHAR2(100) NOT NULL,
    Prenom VARCHAR2(100) NOT NULL,
    DateNaissance DATE NOT NULL,
    Adresse VARCHAR2(255) NOT NULL,
    DateInscription DATE NOT NULL,
    DateResiliation DATE,
    PRIMARY KEY (ID_Membre),
    FOREIGN KEY (ID_Abonnement) REFERENCES Abonnement(ID_Abonnement)
);

CREATE TABLE MoyenPaiement (
    ID_Paiement NUMBER NOT NULL,
    Type_MoyenPaiement VARCHAR2(50) NOT NULL,
    Details VARCHAR2(255),
    PRIMARY KEY (ID_Paiement)
);

CREATE TABLE Equipement (
    ID_Equipement NUMBER NOT NULL,
    Nom VARCHAR2(100) NOT NULL,
    Type_Equipement VARCHAR2(100) NOT NULL,
```

```
    DateAchat DATE NOT NULL,
    Prix NUMBER NOT NULL CHECK (Prix >= 0),
    PRIMARY KEY (ID_Equipement)
);

CREATE TABLE TypeDeCours (
    ID_TypeDeCours NUMBER NOT NULL,
    Type_TypeDeCours VARCHAR2(50) NOT NULL,
    PRIMARY KEY (ID_TypeDeCours)
);

CREATE TABLE Entraineur (
    ID_Entraineur NUMBER NOT NULL,
    Nom VARCHAR2(100) NOT NULL,
    Prenom VARCHAR2(100) NOT NULL,
    Specialite VARCHAR2(100) NOT NULL,
    DateEmbauche DATE NOT NULL,
    PRIMARY KEY (ID_Entraineur)
);

CREATE TABLE Cours (
    ID_Cours NUMBER NOT NULL,
    ID_TypeDeCours NUMBER NOT NULL,
    ID_Entraineur NUMBER NOT NULL,
    Nom VARCHAR2(100) NOT NULL UNIQUE,
    Description VARCHAR2(255),
    PRIMARY KEY (ID_Cours),
    FOREIGN KEY (ID_TypeDeCours) REFERENCES TypeDeCours(ID_TypeDeCours),
    FOREIGN KEY (ID_Entraineur) REFERENCES Entraineur(ID_Entraineur)
);

CREATE TABLE EstMembreDe (
    ID_Salle NUMBER NOT NULL,
    ID_Membre NUMBER NOT NULL,
    FOREIGN KEY (ID_Salle) REFERENCES SalleMusculation(ID_Salle),
    FOREIGN KEY (ID_Membre) REFERENCES Membre(ID_Membre),
    PRIMARY KEY (ID_Salle, ID_Membre)
);

CREATE TABLE PossedePaye (
    ID_Membre NUMBER NOT NULL,
    ID_Paiement NUMBER NOT NULL,
    ID_Abonnement NUMBER NOT NULL,
    Date_PossedePaye DATE NOT NULL,
    FOREIGN KEY (ID_Membre) REFERENCES Membre(ID_Membre),
    FOREIGN KEY (ID_Paiement) REFERENCES MoyenPaiement(ID_Paiement),
    FOREIGN KEY (ID_Abonnement) REFERENCES Abonnement(ID_Abonnement),
    PRIMARY KEY (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
);

CREATE TABLE EstEquipeDe (
    ID_Salle NUMBER NOT NULL,
```

```
    ID_Equipement NUMBER NOT NULL,
    Quantite NUMBER NOT NULL CHECK (Quantite > 0),
    FOREIGN KEY (ID_Salle) REFERENCES SalleMusculation(ID_Salle),
    FOREIGN KEY (ID_Equipement) REFERENCES Equipement(ID_Equipement),
    PRIMARY KEY (ID_Salle, ID_Equipement)
);

CREATE TABLE SeTientDansPropose (
    ID_Salle NUMBER NOT NULL,
    ID_Cours NUMBER NOT NULL,
    HoraireDebut TIMESTAMP NOT NULL,
    HoraireFin TIMESTAMP NOT NULL,
    Jour DATE NOT NULL,
    FOREIGN KEY (ID_Salle) REFERENCES SalleMusculation(ID_Salle),
    FOREIGN KEY (ID_Cours) REFERENCES Cours(ID_Cours),
    PRIMARY KEY (ID_Salle, ID_Cours, Jour, HoraireDebut)
);

CREATE TABLE Participe (
    ID_Membre NUMBER NOT NULL,
    ID_Cours NUMBER NOT NULL,
    Date_Participe DATE NOT NULL,
    FOREIGN KEY (ID_Membre) REFERENCES Membre(ID_Membre),
    FOREIGN KEY (ID_Cours) REFERENCES Cours(ID_Cours),
    PRIMARY KEY (ID_Membre, ID_Cours, Date_Participe)
);
```

B. Jeu de données

Sous Oracle Live, nous avons créé et inséré ce petit jeu de données pour tester nos tables, requêtes, vues et triggers :

```
INSERT ALL
    INTO SalleMusculation (ID_Salle, Nom, Adresse, DateOuverture, SurfaceTotale,
HoraireOuverture, HoraireFermeture)
    VALUES (1, 'Fitness Paradise', '123 Avenue de la Forme, Paris', TO_DATE('2022-01-
01', 'YYYY-MM-DD'), 600, TO_TIMESTAMP('2022-01-01 08:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2022-01-01 22:00:00', 'YYYY-MM-DD HH24:MI:SS'))
    INTO SalleMusculation (ID_Salle, Nom, Adresse, DateOuverture, SurfaceTotale,
HoraireOuverture, HoraireFermeture)
    VALUES (2, 'Iron Strong Gym', '456 Rue de la Musculation, Lyon', TO_DATE('2021-05-
15', 'YYYY-MM-DD'), 800, TO_TIMESTAMP('2021-05-15 06:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2021-05-15 21:30:00', 'YYYY-MM-DD HH24:MI:SS'))
    INTO SalleMusculation (ID_Salle, Nom, Adresse, DateOuverture, SurfaceTotale,
HoraireOuverture, HoraireFermeture)
    VALUES (3, 'FitZone Center', '789 Boulevard Fitness, Marseille', TO_DATE('2020-09-
10', 'YYYY-MM-DD'), 500, TO_TIMESTAMP('2020-09-10 07:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2020-09-10 23:00:00', 'YYYY-MM-DD HH24:MI:SS'))
    INTO SalleMusculation (ID_Salle, Nom, Adresse, DateOuverture, SurfaceTotale,
HoraireOuverture, HoraireFermeture)
    VALUES (4, 'PowerLift Club', '987 Rue de la Force, Bordeaux', TO_DATE('2021-03-
20', 'YYYY-MM-DD'), 700, TO_TIMESTAMP('2021-03-20 09:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2021-03-20 20:00:00', 'YYYY-MM-DD HH24:MI:SS'))
```

```
    INTO SalleMusculation (ID_Salle, Nom, Adresse, DateOuverture, SurfaceTotale,
HoraireOuverture, HoraireFermeture)
    VALUES (5, 'FlexFit Studio', '321 Rue de Flexibilité, Nice', TO_DATE('2023-02-05',
'YYYY-MM-DD'), 450, TO_TIMESTAMP('2023-02-05 10:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-02-05 19:00:00', 'YYYY-MM-DD HH24:MI:SS'))
SELECT * FROM dual;

INSERT ALL
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (1, 'À la semaine', 40.00, TO_DATE('2023-01-15', 'YYYY-MM-DD'),
TO_DATE('2023-01-22', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (2, 'Mensuel', 30.00, TO_DATE('2023-02-01', 'YYYY-MM-DD'), TO_DATE('2023-
02-28', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (3, 'Annuel', 20.00, TO_DATE('2023-03-10', 'YYYY-MM-DD'), TO_DATE('2024-03-
09', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (4, 'À la semaine', 40.00, TO_DATE('2023-04-05', 'YYYY-MM-DD'),
TO_DATE('2023-04-12', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (5, 'Mensuel', 30.00, TO_DATE('2023-05-20', 'YYYY-MM-DD'), TO_DATE('2023-
06-19', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (6, 'Annuel', 20.00, TO_DATE('2023-06-01', 'YYYY-MM-DD'), TO_DATE('2024-05-
31', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (7, 'À la semaine', 40.00, TO_DATE('2023-07-08', 'YYYY-MM-DD'),
TO_DATE('2023-07-15', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (8, 'Mensuel', 30.00, TO_DATE('2023-08-12', 'YYYY-MM-DD'), TO_DATE('2023-
09-11', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (9, 'Annuel', 20.00, TO_DATE('2023-09-25', 'YYYY-MM-DD'), TO_DATE('2024-09-
24', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (10, 'À la semaine', 40.00, TO_DATE('2023-10-15', 'YYYY-MM-DD'),
TO_DATE('2023-10-22', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (11, 'Mensuel', 30.00, TO_DATE('2023-11-05', 'YYYY-MM-DD'), TO_DATE('2023-
12-04', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (12, 'Annuel', 20.00, TO_DATE('2023-12-18', 'YYYY-MM-DD'), TO_DATE('2024-
12-17', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (13, 'À la semaine', 40.00, TO_DATE('2024-01-10', 'YYYY-MM-DD'),
TO_DATE('2024-01-17', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (14, 'Mensuel', 30.00, TO_DATE('2024-02-15', 'YYYY-MM-DD'), TO_DATE('2024-
03-14', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (15, 'Annuel', 20.00, TO_DATE('2024-03-28', 'YYYY-MM-DD'), TO_DATE('2025-
```

```
03-27', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (16, 'À la semaine', 40.00, TO_DATE('2024-04-05', 'YYYY-MM-DD'),
TO_DATE('2024-04-12', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (17, 'Mensuel', 30.00, TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2024-
06-09', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (18, 'Annuel', 20.00, TO_DATE('2024-06-20', 'YYYY-MM-DD'), TO_DATE('2025-
06-19', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (19, 'À la semaine', 40.00, TO_DATE('2024-07-05', 'YYYY-MM-DD'),
TO_DATE('2024-07-12', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (20, 'Mensuel', 30.00, TO_DATE('2024-08-10', 'YYYY-MM-DD'), TO_DATE('2024-
09-09', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (21, 'À la semaine', 40.00, TO_DATE('2024-01-15', 'YYYY-MM-DD'),
TO_DATE('2024-01-22', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (22, 'Mensuel', 30.00, TO_DATE('2024-02-01', 'YYYY-MM-DD'), TO_DATE('2024-
02-29', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (23, 'Annuel', 20.00, TO_DATE('2024-03-10', 'YYYY-MM-DD'), TO_DATE('2025-
03-09', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (24, 'À la semaine', 40.00, TO_DATE('2024-04-05', 'YYYY-MM-DD'),
TO_DATE('2024-04-12', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (25, 'Mensuel', 30.00, TO_DATE('2024-05-20', 'YYYY-MM-DD'), TO_DATE('2024-
06-19', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (26, 'Annuel', 20.00, TO_DATE('2024-06-01', 'YYYY-MM-DD'), TO_DATE('2025-
05-31', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (27, 'À la semaine', 40.00, TO_DATE('2024-07-08', 'YYYY-MM-DD'),
TO_DATE('2024-07-15', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (28, 'Mensuel', 30.00, TO_DATE('2024-08-12', 'YYYY-MM-DD'), TO_DATE('2024-
09-11', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (29, 'Annuel', 20.00, TO_DATE('2024-09-25', 'YYYY-MM-DD'), TO_DATE('2025-
09-24', 'YYYY-MM-DD'))
    INTO Abonnement (ID_Abonnement, Type_Abonnement, Prix, DateDebut, DateFin)
    VALUES (30, 'À la semaine', 40.00, TO_DATE('2024-10-15', 'YYYY-MM-DD'),
TO_DATE('2024-10-22', 'YYYY-MM-DD'))
SELECT * FROM dual;

INSERT ALL
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (1, 1, 'Dupont', 'Alice', TO_DATE('1990-05-15', 'YYYY-MM-DD'), '123 Rue de
```

```
la Forme, Paris', TO_DATE('2022-01-01', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (2, 2, 'Martin', 'Bob', TO_DATE('1985-08-22', 'YYYY-MM-DD'), '456 Avenue Musculaire, Lyon', TO_DATE('2021-06-10', 'YYYY-MM-DD'), TO_DATE('2023-01-15', 'YYYY-MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (3, 3, 'Lefevre', 'Claire', TO_DATE('1993-03-08', 'YYYY-MM-DD'), '789 Boulevard Fitness, Marseille', TO_DATE('2020-11-20', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (4, 4, 'Beaulieu', 'David', TO_DATE('1988-12-03', 'YYYY-MM-DD'), '987 Rue Force, Bordeaux', TO_DATE('2021-04-05', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (5, 5, 'Roux', 'Sophie', TO_DATE('1995-07-19', 'YYYY-MM-DD'), '321 Rue Flexibilite, Nice', TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-06-30', 'YYYY-MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (6, 6, 'Girard', 'Thomas', TO_DATE('1992-09-18', 'YYYY-MM-DD'), '456 Rue Musculaire, Lyon', TO_DATE('2022-02-15', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (7, 7, 'Lemoine', 'Laura', TO_DATE('1991-07-25', 'YYYY-MM-DD'), '789 Boulevard Fitness, Marseille', TO_DATE('2020-12-08', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (8, 8, 'Morel', 'Alexandre', TO_DATE('1987-04-30', 'YYYY-MM-DD'), '987 Rue Force, Bordeaux', TO_DATE('2021-05-20', 'YYYY-MM-DD'), TO_DATE('2022-10-15', 'YYYY-MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (9, 9, 'Lefevre', 'Julie', TO_DATE('1996-01-12', 'YYYY-MM-DD'), '321 Rue Flexibilite, Nice', TO_DATE('2023-03-15', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (10, 10, 'Roy', 'Luc', TO_DATE('1985-11-05', 'YYYY-MM-DD'), '234 Avenue Entraînement, Toulouse', TO_DATE('2022-07-01', 'YYYY-MM-DD'), TO_DATE('2022-12-31', 'YYYY-MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (11, 11, 'Boucher', 'Marie', TO_DATE('1993-06-28', 'YYYY-MM-DD'), '876 Rue Renforcement, Lille', TO_DATE('2020-09-22', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (12, 12, 'Gauthier', 'Nicolas', TO_DATE('1988-03-14', 'YYYY-MM-DD'), '567 Avenue Forme, Strasbourg', TO_DATE('2021-11-10', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse, DateInscription, DateResiliation)
    VALUES (13, 13, 'Leroux', 'Sophie', TO_DATE('1990-08-09', 'YYYY-MM-DD'), '123 Rue
```

```
Gym, Nantes', TO_DATE('2022-03-10', 'YYYY-MM-DD'), TO_DATE('2023-05-01', 'YYYY-MM-
DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (14, 14, 'Coulombe', 'Pierre', TO_DATE('1986-12-17', 'YYYY-MM-DD'), '345
Boulevard Renforcement, Montpellier', TO_DATE('2022-05-05', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (15, 15, 'Poirier', 'Celine', TO_DATE('1995-04-02', 'YYYY-MM-DD'), '678 Rue
Fitness, Rennes', TO_DATE('2023-01-15', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (16, 16, 'Fournier', 'Mathieu', TO_DATE('1989-10-11', 'YYYY-MM-DD'), '987
Avenue Sport, Nice', TO_DATE('2021-08-12', 'YYYY-MM-DD'), TO_DATE('2022-02-28', 'YYYY-
MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (17, 17, 'Lavoie', 'Isabelle', TO_DATE('1994-03-25', 'YYYY-MM-DD'), '456
Rue Exercice, Lyon', TO_DATE('2023-02-10', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (18, 18, 'Dubois', 'Antoine', TO_DATE('1987-06-20', 'YYYY-MM-DD'), '789
Boulevard Sport, Marseille', TO_DATE('2020-10-05', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (19, 19, 'Renaud', 'Camille', TO_DATE('1991-02-08', 'YYYY-MM-DD'), '543 Rue
Musculation, Bordeaux', TO_DATE('2021-07-15', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (20, 20, 'Martin', 'Philippe', TO_DATE('1996-09-14', 'YYYY-MM-DD'), '876
Avenue Renforcement, Paris', TO_DATE('2022-04-20', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (21, 21, 'Gagnon', 'emilie', TO_DATE('1993-11-02', 'YYYY-MM-DD'), '345 Rue
Entraînement, Toulouse', TO_DATE('2022-11-01', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (22, 22, 'Lemieux', 'Gabriel', TO_DATE('1986-04-18', 'YYYY-MM-DD'), '678
Avenue Fitness, Nice', TO_DATE('2023-05-10', 'YYYY-MM-DD'), TO_DATE('2023-10-31',
'YYYY-MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (23, 23, 'Bergeron', 'Julien', TO_DATE('1990-07-22', 'YYYY-MM-DD'), '123
Rue Renforcement, Lyon', TO_DATE('2022-02-28', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (24, 24, 'Morin', 'Melanie', TO_DATE('1989-01-15', 'YYYY-MM-DD'), '456
Avenue Musculaire, Bordeaux', TO_DATE('2022-08-15', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (25, 25, 'Perron', 'Vincent', TO_DATE('1996-06-08', 'YYYY-MM-DD'), '789
Boulevard Forme, Marseille', TO_DATE('2021-05-05', 'YYYY-MM-DD'), TO_DATE('2022-04-
```

```
30', 'YYYY-MM-DD'))
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (26, 26, 'Caron', 'Stephanie', TO_DATE('1992-03-20', 'YYYY-MM-DD'), '987
Rue Sport, Lille', TO_DATE('2022-10-10', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (27, 27, 'Dion', 'Marc', TO_DATE('1987-08-12', 'YYYY-MM-DD'), '321 Rue
Exercice, Nantes', TO_DATE('2020-12-15', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (28, 28, 'Gosselin', 'Valerie', TO_DATE('1991-05-28', 'YYYY-MM-DD'), '543
Avenue Musculation, Rennes', TO_DATE('2023-02-01', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (29, 29, 'Simard', 'etienne', TO_DATE('1988-10-03', 'YYYY-MM-DD'), '876
Boulevard Fitness, Strasbourg', TO_DATE('2022-04-10', 'YYYY-MM-DD'), NULL)
    INTO Membre (ID_Membre, ID_Abonnement, Nom, Prenom, DateNaissance, Adresse,
DateInscription, DateResiliation)
    VALUES (30, 30, 'Larose', 'Nathalie', TO_DATE('1994-09-17', 'YYYY-MM-DD'), '234
Rue Sport, Montpellier', TO_DATE('2021-09-22', 'YYYY-MM-DD'), TO_DATE('2023-03-31',
'YYYY-MM-DD'))
SELECT * FROM dual;

INSERT ALL
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (1, 'Carte Bancaire', 'Card Type: Visa, Card Number: 1234-5678-9012-3456,
Expiry Date: 05/24')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (2, 'Virement', 'Account Holder: John Doe, IBAN: FR76 3000 1234 5678 9012
3456 789, BIC: BNPAFRPPXXX')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (3, 'Paypal', 'Account Email: john.doe@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (4, 'Chèque', 'Payable to: Fitness Center, Address: 123 Gym Street, City:
Fitnessville')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (5, 'Carte Bancaire', 'Card Type: MasterCard, Card Number: 9876-5432-1098-
7654, Expiry Date: 09/23')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (6, 'Virement', 'Account Holder: Jane Smith, IBAN: FR76 3000 5678 9012 3456
7891 234, BIC: BNPAFRPPYYY')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (7, 'Paypal', 'Account Email: jane.smith@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (8, 'Chèque', 'Payable to: Fitness Center, Address: 456 Workout Avenue,
City: Fitcity')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (9, 'Carte Bancaire', 'Card Type: American Express, Card Number: 5678-9012-
3456-7890, Expiry Date: 12/25')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (10, 'Virement', 'Account Holder: Robert Johnson, IBAN: FR76 3000 9012 3456
```

```
7891 2345 678, BIC: BNPAFRPPZZZ')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (11, 'Paypal', 'Account Email: robert.johnson@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (12, 'Chèque', 'Payable to: Fitness Center, Address: 789 Exercise Street,
City: Fitland')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (13, 'Carte Bancaire', 'Card Type: Discover, Card Number: 9012-3456-7890-
1234, Expiry Date: 03/22')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (14, 'Virement', 'Account Holder: Sarah Davis, IBAN: FR76 3000 3456 7891
2345 6789 012, BIC: BNPAFRPPAAA')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (15, 'Paypal', 'Account Email: sarah.davis@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (16, 'Chèque', 'Payable to: Fitness Center, Address: 234 Strength Avenue,
City: Fitville')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (17, 'Carte Bancaire', 'Card Type: Diners Club, Card Number: 3456-7890-
1234-5678, Expiry Date: 06/23')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (18, 'Virement', 'Account Holder: Michael White, IBAN: FR76 3000 7891 2345
6789 0123 456, BIC: BNPAFRPPBBB')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (19, 'Paypal', 'Account Email: michael.white@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (20, 'Chèque', 'Payable to: Fitness Center, Address: 567 Health Street,
City: Fitness City')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (21, 'Carte Bancaire', 'Card Type: Visa, Card Number: 5678-1234-9012-3456,
Expiry Date: 08/23')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (22, 'Virement', 'Account Holder: Jessica Miller, IBAN: FR76 3000 2345 6789
0123 4567 890, BIC: BNPAFRPPCCC')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (23, 'Paypal', 'Account Email: jessica.miller@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (24, 'Chèque', 'Payable to: Fitness Center, Address: 789 Fitness Street,
City: Bodyville')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (25, 'Carte Bancaire', 'Card Type: Maestro, Card Number: 7890-1234-5678-
9012, Expiry Date: 02/24')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (26, 'Virement', 'Account Holder: William Taylor, IBAN: FR76 3000 3456 7890
1234 5678 901, BIC: BNPAFRPPDDD')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (27, 'Paypal', 'Account Email: william.taylor@example.com')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (28, 'Chèque', 'Payable to: Fitness Center, Address: 123 Health Avenue,
City: Wellness City')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (29, 'Carte Bancaire', 'Card Type: JCB, Card Number: 3456-9012-1234-5678,
```

```
Expiry Date: 11/22')
    INTO MoyenPaiement (ID_Paiement, Type_MoyenPaiement, Details)
    VALUES (30, 'Virement', 'Account Holder: Emily Wilson, IBAN: FR76 3000 4567 8901
2345 6789 012, BIC: BNPAFRPPEEE')
SELECT * FROM dual;

INSERT ALL
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (1, 'Seated leg press', 'machine', TO_DATE('2022-01-01', 'YYYY-MM-DD'),
700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (2, 'Decline leg press', 'machine', TO_DATE('2022-02-01', 'YYYY-MM-DD'),
700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (3, 'Leg extension', 'machine', TO_DATE('2022-03-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (4, 'Seated leg curl', 'machine', TO_DATE('2022-04-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (5, 'Lying leg curl', 'machine', TO_DATE('2022-05-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (6, 'Shoulder press', 'machine', TO_DATE('2022-06-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (7, 'Converging chest press', 'machine', TO_DATE('2022-07-01', 'YYYY-MM-
DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (8, 'Pec deck', 'machine', TO_DATE('2022-08-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (9, 'Chest press', 'machine', TO_DATE('2022-09-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (10, 'Lat pulldown', 'machine', TO_DATE('2022-10-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (11, 'Seated row', 'machine', TO_DATE('2022-11-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (12, 'Converging seated row', 'machine', TO_DATE('2022-12-01', 'YYYY-MM-
DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (13, 'Preacher curl', 'machine', TO_DATE('2023-01-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (14, 'Seated dips', 'machine', TO_DATE('2023-02-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (15, 'Long barbell', 'poids libre', TO_DATE('2023-03-01', 'YYYY-MM-DD'),
800)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (16, 'Mid barbell', 'poids libre', TO_DATE('2023-04-01', 'YYYY-MM-DD'),
800)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (17, 'Dumbbell (lot)', 'poids libre', TO_DATE('2023-05-01', 'YYYY-MM-DD'),
800)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (18, 'Weight disk (lot)', 'poids libre', TO_DATE('2023-06-01', 'YYYY-MM-
DD'), 800)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
```

```
    VALUES (19, 'Adjustable bench', 'banc et cadres', TO_DATE('2023-07-01', 'YYYY-MM-DD'), 700)
    INTO Equipement (ID_Equipement, Nom, Type_Equipement, DateAchat, Prix)
    VALUES (20, 'Smith machine', 'banc et cadres', TO_DATE('2023-08-01', 'YYYY-MM-DD'), 700)
SELECT * FROM dual;

INSERT ALL
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (1, 'Yoga')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (2, 'Pilates')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (3, 'CrossFit')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (4, 'Abdos Fessier')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (5, 'Spinning')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (6, 'Kickboxing')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (7, 'Danse Fitness')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (8, 'Jumping Jack')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (9, 'Bodybuilding')
    INTO TypeDeCours (ID_TypeDeCours, Type_TypeDeCours)
    VALUES (10, 'HIIT')
SELECT * FROM dual;

INSERT ALL
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (1, 'Colas', 'Stéphanie', 'Spinning', TO_DATE('2023-03-15', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (2, 'Perrin', 'Denis', 'Danse Fitness', TO_DATE('2023-08-09', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (3, 'Leclercq', 'Xavier', 'Abdo Fessier', TO_DATE('2019-07-16', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (4, 'Schneider', 'Hugues', 'Kickboxing', TO_DATE('2018-08-11', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (5, 'Charles', 'Frédéric', 'Bodybuilding', TO_DATE('2020-08-31', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (6, 'Vincent', 'Christiane', 'Jumping Jack', TO_DATE('2014-10-03', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (7, 'Antoine', 'Aimée', 'HIIT', TO_DATE('2014-10-26', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (8, 'Barbier', 'Victor', 'CrossFit', TO_DATE('2018-05-26', 'YYYY-MM-DD'))
```

```
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (9, 'Neveu', 'Capucine', 'Pilates', TO_DATE('2023-06-07', 'YYYY-MM-DD'))
    INTO Entraineur (ID_Entraineur, Nom, Prenom, Specialite, DateEmbauche)
    VALUES (10, 'Gosselin', 'Valentine', 'Yoga', TO_DATE('2020-11-17', 'YYYY-MM-DD'))
SELECT * FROM dual;

INSERT ALL
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (1, 1, 10, 'Yoga Flow', 'Séance de yoga dynamique pour renforcer et
assouplir le corps.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (2, 2, 9, 'Pilates Core', 'Cours de Pilates axé sur le renforcement des
muscles profonds et la stabilité du tronc.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (3, 3, 8, 'CrossFit Challenge', 'Entraînement intensif mêlant différents
exercices pour développer la force et l''endurance.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (4, 4, 3, 'Booty Blast', 'Séance ciblée sur les muscles des fesses et des
jambes pour un renforcement efficace.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (5, 5, 1, 'Spinning Adventure', 'Voyage virtuel à travers différents
terrains en pédalant sur un vélo stationnaire.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (6, 6, 4, 'Kickboxing Cardio', 'Combinaison de mouvements de kickboxing
pour améliorer la condition physique.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (7, 7, 2, 'Dance Fusion', 'Cours de danse énergique combinant différents
styles pour brûler des calories.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (8, 8, 5, 'Jumping Jack Joy', 'Entraînement cardio avec des sauts et des
mouvements de jumping jack.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (9, 9, 7, 'Bodybuilding Basics', 'Introduction aux exercices de musculation
pour le renforcement musculaire.')
    INTO Cours (ID_Cours, ID_TypeDeCours, ID_Entraineur, Nom, Description)
    VALUES (10, 10, 6, 'HIIT Intense', 'Entraînement fractionné de haute intensité
pour améliorer la condition physique globale.')
SELECT * FROM dual;

INSERT ALL
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (1, 1)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (2, 2)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (3, 3)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (4, 4)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (5, 5)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (1, 6)
```

```
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (2, 7)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (3, 8)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (4, 9)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (5, 10)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (1, 11)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (2, 12)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (3, 13)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (4, 14)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (5, 15)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (1, 16)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (2, 17)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (3, 18)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (4, 19)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (5, 20)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (1, 21)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (2, 22)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (3, 23)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (4, 24)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (5, 25)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (1, 26)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (2, 27)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (3, 28)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (4, 29)
    INTO EstMembreDe (ID_Salle, ID_Membre)
    VALUES (5, 30)
SELECT * FROM dual;

INSERT ALL
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
```

```
    VALUES (1, 1, 1, TO_DATE('2023-01-15', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (2, 2, 2, TO_DATE('2023-02-01', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (3, 3, 3, TO_DATE('2023-03-10', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (4, 4, 4, TO_DATE('2023-04-05', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (5, 5, 5, TO_DATE('2023-05-20', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (6, 6, 6, TO_DATE('2023-06-01', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (7, 7, 7, TO_DATE('2023-07-08', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (8, 8, 8, TO_DATE('2023-08-12', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (9, 9, 9, TO_DATE('2023-09-25', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (10, 10, 10, TO_DATE('2023-10-03', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (11, 11, 11, TO_DATE('2023-11-18', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (12, 12, 12, TO_DATE('2024-01-07', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (13, 13, 13, TO_DATE('2024-02-14', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (14, 14, 14, TO_DATE('2024-03-02', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (15, 15, 15, TO_DATE('2024-04-15', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (16, 16, 16, TO_DATE('2024-05-22', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (17, 17, 17, TO_DATE('2024-06-10', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (18, 18, 18, TO_DATE('2024-07-01', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (19, 19, 19, TO_DATE('2024-08-05', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (20, 20, 20, TO_DATE('2024-09-20', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (21, 21, 21, TO_DATE('2024-10-15', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (22, 22, 22, TO_DATE('2024-11-01', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (23, 23, 23, TO_DATE('2024-12-10', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (24, 24, 24, TO_DATE('2025-01-05', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (25, 25, 25, TO_DATE('2025-02-20', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (26, 26, 26, TO_DATE('2025-03-01', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
```

```
    VALUES (27, 27, 27, TO_DATE('2025-04-08', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (28, 28, 28, TO_DATE('2025-05-12', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (29, 29, 29, TO_DATE('2025-06-25', 'YYYY-MM-DD'))
    INTO PossedePaye (ID_Membre, ID_Paiement, ID_Abonnement, Date_PossedePaye)
    VALUES (30, 30, 30, TO_DATE('2025-07-03', 'YYYY-MM-DD'))
SELECT * FROM dual;

-- Équipement pour Salle 1
INSERT ALL
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 1, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 2, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 3, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 4, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 5, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 6, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 7, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 8, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 9, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 10, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 11, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 12, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 13, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 14, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 15, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 16, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 17, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 18, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 19, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (1, 20, 3)
SELECT * FROM dual;
```

```sql
-- Équipement pour Salle 2
INSERT ALL
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 1, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 2, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 3, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 4, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 5, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 6, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 7, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 8, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 9, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 10, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 11, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 12, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 13, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 14, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 15, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 16, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 17, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 18, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 19, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (2, 20, 3)
SELECT * FROM dual;

-- Équipement pour Salle 3
INSERT ALL
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 1, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 2, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
```

```
    VALUES (3, 3, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 4, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 5, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 6, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 7, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 8, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 9, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 10, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 11, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 12, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 13, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 14, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 15, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 16, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 17, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 18, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 19, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (3, 20, 2)
SELECT * FROM dual;

-- Équipement pour Salle 4
INSERT ALL
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 1, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 2, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 3, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 4, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 5, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 6, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
```

```
    VALUES (4, 7, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 8, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 9, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 10, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 11, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 12, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 13, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 14, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 15, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 16, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 17, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 18, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 19, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (4, 20, 2)
SELECT * FROM dual;

-- Équipement pour Salle 5
INSERT ALL
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 1, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 2, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 3, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 4, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 5, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 6, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 7, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 8, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 9, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 10, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
```

```
    VALUES (5, 11, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 12, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 13, 2)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 14, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 15, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 16, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 17, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 18, 4)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 19, 3)
    INTO EstEquipeDe (ID_Salle, ID_Equipement, Quantite)
    VALUES (5, 20, 3)
SELECT * FROM dual;

INSERT ALL
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (1, 1, TO_TIMESTAMP('2023-01-15 10:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-01-15 11:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-01-15',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (2, 2, TO_TIMESTAMP('2023-02-01 14:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-02-01 15:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-02-01',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (3, 3, TO_TIMESTAMP('2023-03-10 18:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-03-10 19:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-03-10',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (4, 4, TO_TIMESTAMP('2023-04-05 19:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-04-05 20:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-04-05',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (5, 5, TO_TIMESTAMP('2023-05-20 12:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-05-20 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-05-20',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (1, 6, TO_TIMESTAMP('2023-06-01 16:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-06-01 18:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-06-01',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (2, 7, TO_TIMESTAMP('2023-07-08 17:00:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-07-08 18:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-07-08',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (3, 8, TO_TIMESTAMP('2023-08-12 20:00:00', 'YYYY-MM-DD HH24:MI:SS'),
```

```
TO_TIMESTAMP('2023-08-12 21:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-08-12',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (4, 9, TO_TIMESTAMP('2023-09-25 14:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-09-25 16:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-09-25',
'YYYY-MM-DD'))
    INTO SeTientDansPropose (ID_Salle, ID_Cours, HoraireDebut, HoraireFin, Jour)
    VALUES (5, 10, TO_TIMESTAMP('2023-10-03 15:30:00', 'YYYY-MM-DD HH24:MI:SS'),
TO_TIMESTAMP('2023-10-03 17:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2023-10-03',
'YYYY-MM-DD'))
SELECT * FROM dual;
--

INSERT ALL
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (1, 1, TO_DATE('2023-01-15', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (1, 2, TO_DATE('2023-02-01', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (2, 1, TO_DATE('2023-02-15', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (2, 2, TO_DATE('2023-02-01', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (3, 3, TO_DATE('2023-03-10', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (3, 6, TO_DATE('2023-03-25', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (4, 4, TO_DATE('2023-04-05', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (5, 5, TO_DATE('2023-05-20', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (6, 6, TO_DATE('2023-06-01', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (7, 7, TO_DATE('2023-07-08', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (8, 8, TO_DATE('2023-08-12', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (9, 9, TO_DATE('2023-09-25', 'YYYY-MM-DD'))
    INTO Participe (ID_Membre, ID_Cours, Date_Participe)
    VALUES (10, 10, TO_DATE('2023-10-03', 'YYYY-MM-DD'))
SELECT * FROM dual;
```

Pour ce qui est de la méthode avec `SQL*Load`, nous avons rencontré des problèmes avec la machine virtuelle que nous avons pas réussi à corriger. Cependant, nous avons créé un script Python pour générer des données et nous le mentionnons à titre indicatif à la fin de ce compte rendu.

C. Manipulation des données

1. Quel est le nombre total de membres ?

```
SELECT COUNT(ID_Membre) AS Total_Membres
FROM Membre;
```

2. Quel est le revenu moyen des abonnements ?

```
SELECT AVG(Prix) AS Revenu_Moyen_Abonnements
FROM Abonnement;
```

3. Quel est le nombre d'équipements par salle ?

```
SELECT ID_Salle, SUM(Quantite) AS Total_Equipements
FROM EstEquipeDe
GROUP BY ID_Salle;
```

4. Quels sont tous les membres avec un abonnement actif ?

```
SELECT M.ID_Membre, M.Nom, M.Prenom
FROM Membre M
JOIN Abonnement A ON M.ID_Abonnement = A.ID_Abonnement
WHERE A.DateFin IS NULL OR A.DateFin > SYSDATE;
```

5. Quels sont le nom et le prénom des membres ?

```
SELECT Nom, Prenom
FROM Membre;
```

6. Quel abonnement est lié à quel membre ?

```
SELECT M.ID_Membre, M.Nom, M.Prenom, A.Type_Abonnement, A.Prix
FROM Membre M
JOIN Abonnement A ON M.ID_Abonnement = A.ID_Abonnement;
```

7. Quels cours sont liés à quel entraîneur ?

```
SELECT C.ID_Cours, C.Nom, E.Nom AS Nom_Entraineur, E.Prenom AS
Prenom_Entraineur
FROM Cours C
JOIN Entraineur E ON C.ID_Entraineur = E.ID_Entraineur;
```

8. Quels équipements sont liés à quelle salle ?

```
SELECT E.ID_Equipement, E.Nom, S.ID_Salle, S.Nom AS Nom_Salle
FROM Equipement E
JOIN EstEquipeDe EE ON E.ID_Equipement = EE.ID_Equipement
JOIN SalleMusculation S ON EE.ID_Salle = S.ID_Salle;
```

9. Quels sont les membres inscrits à tous les cours proposés par un certain entraîneur (remplacer `[EntraineurID]` par l'ID de l'entraineur qu'on veut) ?

```
SELECT M.ID_Membre, M.Nom, M.Prénom
FROM Membre M
WHERE NOT EXISTS (
    SELECT C.ID_Cours
    FROM Cours C
    WHERE C.ID_Entraineur = [EntraineurID] AND NOT EXISTS (
        SELECT P.ID_Cours
        FROM Participe P
```

```
        WHERE P.ID_Membre = M.ID_Membre AND P.ID_Cours = C.ID_Cours
    )
);
```

10. Quels sont les membres qui sont à la fois dans des cours de yoga et de Pilates ?

```
-- Membres dans des cours de cardio
WITH Cardio AS (
    SELECT P.ID_Membre
    FROM Participe P
    JOIN Cours C ON P.ID_Cours = C.ID_Cours
    JOIN TypeDeCours T ON C.ID_TypeDeCours = T.ID_TypeDeCours
    WHERE T.Type_TypeDeCours = 'Yoga'
),
-- Membres dans des cours de musculation
Muscu AS (
    SELECT P.ID_Membre
    FROM Participe P
    JOIN Cours C ON P.ID_Cours = C.ID_Cours
    JOIN TypeDeCours T ON C.ID_TypeDeCours = T.ID_TypeDeCours
    WHERE T.Type_TypeDeCours = 'Pilates'
)
-- Intersection
SELECT M.ID_Membre, M.Nom, M.Prenom
FROM Membre M
WHERE M.ID_Membre IN (SELECT ID_Membre FROM Cardio)
AND M.ID_Membre IN (SELECT ID_Membre FROM Muscu);
```

11. Quels membres ont assisté à la fois à des cours de CrossFit et de Kickboxing le même mois ?

```
-- Membres dans des cours de crossfit
WITH Crossfit AS (
    SELECT P.ID_Membre, EXTRACT(MONTH FROM P.Date_Participe) AS Mois
    FROM Participe P
    JOIN Cours C ON P.ID_Cours = C.ID_Cours
    JOIN TypeDeCours T ON C.ID_TypeDeCours = T.ID_TypeDeCours
    WHERE T.Type_TypeDeCours = 'CrossFit'
),
-- Membres dans des cours de kickboxing
Kickboxing AS (
    SELECT P.ID_Membre, EXTRACT(MONTH FROM P.Date_Participe) AS Mois
    FROM Participe P
    JOIN Cours C ON P.ID_Cours = C.ID_Cours
    JOIN TypeDeCours T ON C.ID_TypeDeCours = T.ID_TypeDeCours
    WHERE T.Type_TypeDeCours = 'Kickboxing'
)
-- Intersection
SELECT M.ID_Membre, M.Nom, M.Prenom
FROM Membre M
WHERE EXISTS (
    SELECT 1
```

```
    FROM Crossfit C
    JOIN Kickboxing K ON C.ID_Membre = K.ID_Membre AND C.Mois = K.Mois
    WHERE M.ID_Membre = C.ID_Membre
);
```

12. Quelle est la liste complète des membres actifs et des entraîneurs ?

```
SELECT ID_Membre AS ID, Nom, Prenom, 'Membre' AS Type
FROM Membre
WHERE ID_Abonnement IN (SELECT ID_Abonnement FROM Abonnement WHERE DateFin IS
NULL OR DateFin > SYSDATE)
UNION
SELECT ID_Entraineur AS ID, Nom, Prenom, 'Entraineur' AS Type
FROM Entraineur;
```

13. Quels équipements sont disponibles dans certaines salles mais pas dans d'autres (remplacer `[SalleAID]` et `[SalleBID]` par les ID de salle voulus) ?

```
-- Équipements dans la salle A
WITH SalleA AS (
    SELECT E.ID_Equipement, E.Nom
    FROM EstEquipeDe EE
    JOIN Equipement E ON EE.ID_Equipement = E.ID_Equipement
    WHERE EE.ID_Salle = [SalleAID]
),
-- Équipements dans la salle B
SalleB AS (
    SELECT E.ID_Equipement, E.Nom
    FROM EstEquipeDe EE
    JOIN Equipement E ON EE.ID_Equipement = E.ID_Equipement
    WHERE EE.ID_Salle = [SalleBID]
)
-- Différence
SELECT ID_Equipement, Nom
FROM SalleA
MINUS
SELECT ID_Equipement, Nom
FROM SalleB;
```

14. Quels sont tous les entraîneurs et les cours qu'ils donnent (y compris ceux qui n'ont pas encore de cours attribués) ?

```
SELECT E.ID_Entraineur, E.Nom AS Nom_Entraineur, E.Prenom AS Prenom_Entraineur,
C.ID_Cours, C.Nom AS Nom_Cours
FROM Entraineur E
LEFT JOIN Cours C ON E.ID_Entraineur = C.ID_Entraineur;
```

15. Que donne la jointure des membres, abonnements, paiements, et cours pour obtenir un rapport détaillé sur les activités des membres ?

```
SELECT M.ID_Membre, M.Nom, M.Prenom, A.Type_Abonnement, A.Prix,
PP.Date_PossedePaye, C.Nom AS Nom_Cours
FROM Membre M
```

```
JOIN Abonnement A ON M.ID_Abonnement = A.ID_Abonnement
JOIN PossedePaye PP ON M.ID_Membre = PP.ID_Membre
JOIN Participe P ON M.ID_Membre = P.ID_Membre
JOIN Cours C ON P.ID_Cours = C.ID_Cours;
```

16. Quels sont les membres qui ont suivi tous les types de cours proposés ?

```
SELECT M.ID_Membre, M.Nom, M.Prenom
FROM Membre M
WHERE NOT EXISTS (
    SELECT T.ID_TypeDeCours
    FROM TypeDeCours T
    WHERE NOT EXISTS (
        SELECT P.ID_Cours
        FROM Participe P
        JOIN Cours C ON P.ID_Cours = C.ID_Cours
        WHERE P.ID_Membre = M.ID_Membre AND C.ID_TypeDeCours = T.ID_TypeDeCours
    )
);
```

D. Vues et rôles

- Vue du nombre total d'équipements par salle de musculation :

```
  CREATE VIEW EquipementsParSalle AS
  SELECT ID_Salle, COUNT(ID_Equipement) AS NombreEquipements
  FROM EstEquipeDe
  GROUP BY ID_Salle;
```

- Vue des revenus moyens par type d'abonnement :

```
  CREATE VIEW RevenuMoyenAbonnement AS
  SELECT Type_Abonnement, AVG(Prix) AS RevenuMoyen
  FROM Abonnement
  GROUP BY Type_Abonnement;
```

- Vue des membres et de leur dernier paiement :

```
  CREATE VIEW DernierPaiementMembre AS
  SELECT M.ID_Membre, M.Nom, M.Prenom, MAX(P.Date_PossedePaye) AS
DernierPaiement
  FROM Membre M
  JOIN PossedePaye P ON M.ID_Membre = P.ID_Membre
  GROUP BY M.ID_Membre, M.Nom, M.Prenom;
```

- Vue des cours et du nombre de participants :

```
  CREATE VIEW ParticipantsParCours AS
  SELECT C.ID_Cours, C.Nom, COUNT(P.ID_Membre) AS NombreParticipants
  FROM Cours C
  JOIN Participe P ON C.ID_Cours = P.ID_Cours
  GROUP BY C.ID_Cours, C.Nom;
```

- Vue des entraîneurs et du nombre de cours qu'ils donnent :

```
  CREATE VIEW CoursParEntraineur AS
  SELECT E.ID_Entraineur, E.Nom, E.Prenom, COUNT(C.ID_Cours) AS NombreCours
  FROM Entraineur E
  JOIN Cours C ON E.ID_Entraineur = C.ID_Cours
  GROUP BY E.ID_Entraineur, E.Nom, E.Prenom;
```

- Rôle `Administrateur` (accès complet à toutes les données et fonctionnalités de la base de données, peut ajouter, modifier, supprimer et lire toutes les données, ainsi que gérer les utilisateurs et les rôles) :

```
  CREATE ROLE Administrateur;
  GRANT ALL PRIVILEGES ON DATABASE MaBaseDeDonnees TO Administrateur;
```

- Rôle `GestionnaireDeSalle` (peut gérer les informations relatives à une salle de musculation spécifique, y compris l'ajout, la modification et la suppression d'équipements et de cours, ainsi que les paiements effectués) :

```
  CREATE ROLE GestionnaireDeSalle;
  GRANT SELECT, INSERT, UPDATE, DELETE ON SalleMusculation, Equipement, Cours
TO GestionnaireDeSalle;
  GRANT SELECT, INSERT, UPDATE ON Paiements, MoyenPaiement TO
GestionnaireDeSalle;
```

- Rôle `Entraineur` (peut ajouter, modifier et supprimer des cours qu'il donne) :

```
  CREATE ROLE Entraineur;
  GRANT SELECT, INSERT, UPDATE, DELETE ON Cours TO Entraineur;
```

- Rôle `Membre` (accès en lecture seule à certaines informations pertinentes pour un membre de la salle de musculation, comme les détails des cours auxquels il participe, ou la possibilité de consulter ses moyens de paiements et de transactions, et les mettre à jour au besoin) :

```
  CREATE ROLE Membre;
  GRANT SELECT ON Cours, Participe TO Membre;
  GRANT SELECT, INSERT, UPDATE ON MoyenPaiement, Paiements TO Membre WHERE
MembreID = USER_ID();
```

E. Triggers

- Trigger `CheckDateOuverture` (avant d'insérer ou de mettre à jour une salle de musculation, ce trigger vérifie si la date d'ouverture indiquée est postérieure à la date actuelle) :

```
  CREATE OR REPLACE TRIGGER CheckDateOuverture
  BEFORE INSERT OR UPDATE ON SalleMusculation
  FOR EACH ROW
  BEGIN
      IF :NEW.DateOuverture > SYSDATE THEN
          RAISE_APPLICATION_ERROR(-20001, 'La date d''ouverture doit être
inférieure ou égale à la date actuelle.');
```

```
    END IF;
END;
```

- Trigger `CheckDateNaissance` (avant d'insérer ou de mettre à jour un membre, ce trigger vérifie si la date de naissance indiquée est postérieure à la date actuelle) :

```
CREATE OR REPLACE TRIGGER CheckDateNaissance
BEFORE INSERT OR UPDATE ON Membre
FOR EACH ROW
BEGIN
    IF :NEW.DateNaissance > SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20002, 'La date de naissance doit être
inférieure ou égale à la date actuelle.');
    END IF;
END;
```

- Trigger `CheckEquipementDateAchat` (avant d'insérer ou de mettre à jour un équipement, ce trigger vérifie si la date d'achat indiquée est postérieure à la date actuelle) :

```
CREATE OR REPLACE TRIGGER CheckEquipementDateAchat
BEFORE INSERT OR UPDATE ON Equipement
FOR EACH ROW
BEGIN
    IF :NEW.DateAchat > SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20007, 'La date d''achat de l''équipement
doit être inférieure ou égale à la date actuelle.');
    END IF;
END;
```

- Trigger `CheckEntraineurDateEmbauche` (avant d'insérer ou de mettre à jour un entraîneur, ce trigger vérifie si la date d'embauche indiquée est postérieure à la date actuelle) :

```
CREATE OR REPLACE TRIGGER CheckEntraineurDateEmbauche
BEFORE INSERT OR UPDATE ON Entraineur
FOR EACH ROW
BEGIN
    IF :NEW.DateEmbauche > SYSDATE THEN
        RAISE_APPLICATION_ERROR(-20008, 'La date d''embauche de l''entraineur
doit être inférieure ou égale à la date actuelle.');
    END IF;
END;
```

- Trigger `CheckMemberAge` (avant d’insérer un nouveau membre, s’assure que le membre est âgé d'au moins 18 ans au moment de son inscription) :

```
CREATE OR REPLACE TRIGGER CheckMemberAge
BEFORE INSERT ON Membre
FOR EACH ROW
BEGIN
    IF MONTHS_BETWEEN(SYSDATE, :NEW.DateNaissance)/12 < 18 THEN
```

```
          RAISE_APPLICATION_ERROR(-20001, 'Le membre doit être âgé de 18 ans ou
plus.');
      END IF;
  END;
```

- Trigger `CheckResiliationDate` (avant de mettre à jour un membre, vérifie si la date de résiliation spécifiée est postérieure à la date d'inscription du membre) :

```
  CREATE OR REPLACE TRIGGER CheckResiliationDate
  BEFORE UPDATE OF DateResiliation ON Membre
  FOR EACH ROW
  BEGIN
      IF :NEW.DateResiliation IS NOT NULL AND :NEW.DateResiliation <=
:OLD.DateInscription THEN
          RAISE_APPLICATION_ERROR(-20002, 'La date de résiliation doit être
postérieure à la date d''inscription.');
      END IF;
  END;
```

Les trois triggers suivants posent problème avec nos tables et nos données de test, probablement car nous n'avons pas utilisé le type TIMESTAMP correctement pour les horaires, ainsi ils ne fonctionnent pas et nous n'avons pas le temps de les corriger. Nous les mentionnons à titre indicatif.

- Trigger `CheckCoursHoraireDebut` (après avoir inséré ou mis à jour un cours dans la table `SeTientDansPropose`, ce trigger vérifie si l'horaire de début du cours est postérieur à l'horaire d'ouverture de la salle de musculation associée) :

```
  CREATE OR REPLACE TRIGGER CheckCoursHoraireDebut
  AFTER INSERT OR UPDATE ON SeTientDansPropose
  FOR EACH ROW
  DECLARE
      v_HoraireOuverture TIMESTAMP;
  BEGIN
      SELECT HoraireOuverture INTO v_HoraireOuverture FROM SalleMusculation
WHERE ID_Salle = :NEW.ID_Salle;
      IF :NEW.HoraireDebut < v_HoraireOuverture THEN
          RAISE_APPLICATION_ERROR(-20009, 'L''horaire de début du cours doit
être après l''horaire d''ouverture de la salle.');
      END IF;
  END;
```

- Trigger `CheckCoursHoraireFin` (après avoir inséré ou mis à jour un cours dans la table `SeTientDansPropose`, ce trigger vérifie si l'horaire de fin du cours est antérieur à l'horaire de fermeture de la salle de musculation associée) :

```
  CREATE OR REPLACE TRIGGER CheckCoursHoraireFin
  AFTER INSERT OR UPDATE ON SeTientDansPropose
  FOR EACH ROW
  DECLARE
      v_HoraireFermeture TIMESTAMP;
  BEGIN
```

```
      SELECT HoraireFermeture INTO v_HoraireFermeture FROM SalleMusculation
WHERE ID_Salle = :NEW.ID_Salle;
      IF :NEW.HoraireFin > v_HoraireFermeture THEN
          RAISE_APPLICATION_ERROR(-20010, 'L''horaire de fin du cours doit être
avant l''horaire de fermeture de la salle.');
      END IF;
  END;
```

- Trigger `CheckCoursJour` (avant d'insérer un cours dans la table `SeTientDansPropose`, ce trigger vérifie si le jour programmé pour le cours est au moins le lendemain de la date actuelle) :

```
  CREATE OR REPLACE TRIGGER CheckCoursJour
  BEFORE INSERT ON SeTientDansPropose
  FOR EACH ROW
  BEGIN
      IF :NEW.Jour < SYSDATE + 1 THEN
          RAISE_APPLICATION_ERROR(-20011, 'Les cours doivent être créés pour au
moins le jour suivant.');
      END IF;
  END;
```

F. Méta-données

```
-- Liste des triggers pour la table SalleMusculation
SELECT
    SalleMusculation,
    CheckDateOuverture,
    BEFORE EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

-- Liste des triggers pour la table Membre
SELECT
    Membre,
    CheckDateNaissance,
    BEFORE EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

SELECT
    Membre,
    CheckDateInscription,
    AFTER EACH ROW ,
    INSERT OR UPDATE,
FROM
```

```
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

SELECT
    Membre,
    CheckDateResiliation,
    AFTER EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

-- Liste des triggers pour la table Equipement
SELECT
    Equipement,
    CheckEquipementDateAchat,
    BEFORE EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

-- Liste des triggers pour la table Entraineur
SELECT
    Entraineur,
    CheckEntraineurDateEmbauche,
    BEFORE EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

-- Liste des triggers pour la table SeTientDansPropose
SELECT
    SeTientDansPropose,
    CheckCoursHoraireDebut,
    AFTER EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

SELECT
    SeTientDansPropose,
    CheckCoursHoraireFin,
    AFTER EACH ROW ,
    INSERT OR UPDATE,
```

```
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';

SELECT
    SeTientDansPropose,
    CheckCoursJour,
    BEFORE EACH ROW ,
    INSERT OR UPDATE,
FROM
    all_triggers
WHERE
    owner = 'GestionnaireDeSalle';
```

Autre

Ci-après, notre cahier des charges de la partie 1. À noter que notre implémentation diffère légèrement de ce que nous avions fait dans la partie 1 (notamment les triggers), différences liées aux contraintes d'implémentation que nous n'avions pas prévues. Également, comme mentionné à la fin de la partie B., nous joignons le code qui avait pour but de générer les données de test en quantitée massive pour la méthode avec `SQL*LOAD`.

```
from faker import Faker
from io import StringIO
from datetime import timedelta, datetime
import random
import csv

# Initialize Faker to generate data in French
fake = Faker('fr_FR')
# Define the equipment list as provided
equipments_list = [
    ("Seated leg press", "machine", (200, 700)),
    ("Decline leg press", "machine", (200, 700)),
    ("Leg extension", "machine", (200, 700)),
    ("Seated leg curl", "machine", (200, 700)),
    ("Lying leg curl", "machine", (200, 700)),
    ("Shoulder press", "machine", (200, 700)),
    ("Converging chest press", "machine", (200, 700)),
    ("Pec deck", "machine", (200, 700)),
    ("Chest press", "machine", (200, 700)),
    ("Lat pulldown", "machine", (200, 700)),
    ("Seated row", "machine", (200, 700)),
    ("Converging seated row", "machine", (200, 700)),
    ("Preacher curl", "machine", (200, 700)),
    ("Seated dips", "machine", (200, 700)),
    ("Long barbell", "poids libre", (300, 800)),
    ("Mid barbell", "poids libre", (300, 800)),
    ("Dumbbell (lot)", "poids libre", (300, 800)),
    ("Weight disk (lot)", "poids libre", (300, 800)),
    ("Adjustable bench", "banc et cadres", (200, 700)),
```

```
    ("Smith machine", "banc et cadres", (200, 700))
]
# Define the list of course types as provided
types_de_cours_list = [
    "Yoga",
    "Pilates",
    "CrossFit",
    "Abdos Fessier",
    "Spinning",
    "Kickboxing",
    "Danse Fitness",
    "Jumping Jack",
    "Bodybuilding",
    "HIIT"  # High-Intensity Interval Training
]
# Function to generate formatted data for SalleMusculation
def generate_salle_musculation_formatted(n):
    # Create a string buffer to hold the CSV data
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    opening_hours = [f"{hour:02d}:{minute:02d}:00" for hour in range(4, 10) for minute
in range(0, 60, 15)]
    closing_hours = [f"{hour:02d}:{minute:02d}:00" for hour in range(19, 24) for
minute in range(0, 60, 15)]

    for i in range(n):
        id_salle = fake.unique.random_int(min=1, max=10000)
        nom = fake.company()
        adresse = fake.address().replace('\n', ', ')
        date_ouverture = fake.date_between(start_date='-10y',
end_date='today').strftime('%Y-%m-%d')
        surface_totale = random.randint(400, 1000)  # Surface between 400 and 1000 m^2
        horaire_ouverture = random.choice(opening_hours)
        horaire_fermeture = random.choice(closing_hours)

        # Write a row of data in the specified format
        writer.writerow([i+1, id_salle, nom, adresse, date_ouverture, surface_totale,
horaire_ouverture, horaire_fermeture])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()

# Generate formatted entries for 'SalleMusculation'
formatted_data_salle = generate_salle_musculation_formatted(30)
# Function to generate data for Membre
def generate_membres_formatted(n):
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    for i in range(n):
```

```
        id_membre = fake.unique.random_int(min=1, max=10000)
        id_abonnement = fake.random_int(min=1, max=100)
        nom = fake.last_name()
        prenom = fake.first_name()
        date_naissance = fake.date_of_birth(minimum_age=18,
maximum_age=70).strftime('%Y-%m-%d')
        adresse = fake.address().replace('\n', ', ')
        date_inscription = fake.date_between(start_date='-10y',
end_date='today').strftime('%Y-%m-%d')
        # Randomly decide whether to assign a date of resiliation
        assign_resiliation = random.choice([True, False])
        date_resiliation = fake.date_between(start_date='-1m',
end_date='today').strftime('%Y-%m-%d') if assign_resiliation else None

        # Write a row of data in the specified format
        writer.writerow([i+1, id_membre, id_abonnement, nom, prenom, date_naissance,
adresse, date_inscription, date_resiliation or "None"])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()

# Generate formatted entries for 'Membres'
formatted_data_membres = generate_membres_formatted(2100)
# Function to generate data for MoyenPaiement
# Adjusted function to generate formatted data for MoyenPaiement
def generate_moyen_paiement_formatted(n):
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    for i in range(n):
        id_paiement = fake.unique.random_int(min=1, max=10000)
        type_moyen_paiement = random.choice(['Carte Bancaire', 'Virement', 'Paypal',
'Chèque'])
        # Generate credit card details only for 'Carte Bancaire', otherwise set to
None
        details = fake.credit_card_full() if type_moyen_paiement == 'Carte Bancaire'
else None
        details_formatted = details.replace("\n", ", ") if details is not None else
"None"

        # Write a row of data in the specified format
        writer.writerow([i+1, id_paiement, type_moyen_paiement, details_formatted])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()

# Generate 10 formatted entries for 'MoyenPaiement'
formatted_data_moyen_paiement = generate_moyen_paiement_formatted(2100)
# Adjusted function to generate formatted data for Abonnements
def generate_abonnements_formatted(n):
```

```
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    types_prix = {
        'À la semaine': 10,  # 10€/semaine
        'Mensuel': 30,  # 30€/mois
        'Annuel': 240  # 240€/an
    }

    for i in range(n):
        id_abonnement = fake.unique.random_int(min=1, max=10000)
        type_abonnement, prix = random.choice(list(types_prix.items()))
        date_debut = fake.date_between(start_date='-2y',
end_date='today').strftime('%Y-%m-%d')
        # Adjusting the end date calculation to avoid the out-of-range error
        if type_abonnement == 'À la semaine':
            date_fin = (datetime.strptime(date_debut, '%Y-%m-%d') +
timedelta(weeks=1)).strftime('%Y-%m-%d')
        elif type_abonnement == 'Mensuel':
            date_fin = (datetime.strptime(date_debut, '%Y-%m-%d') +
timedelta(weeks=4)).strftime('%Y-%m-%d')
        else:  # Annuel
            date_fin = (datetime.strptime(date_debut, '%Y-%m-%d') +
timedelta(weeks=52)).strftime('%Y-%m-%d')

        # Write a row of data in the specified format
        writer.writerow([i+1, id_abonnement, type_abonnement, prix, date_debut,
date_fin])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()

# Generate 10 formatted entries for 'Abonnements'
formatted_data_abonnements = generate_abonnements_formatted(2100)
# Updated list of equipment names and types with their respective price ranges
# Adjusted function to generate formatted data for Equipement with specific price
ranges
def generate_equipements_formatted(n, equipment_list):
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    for i in range(n):
        # Randomly select an equipment from the list
        name, type_equipement, price_range = random.choice(equipment_list)
        id_equipement = fake.unique.random_int(min=1, max=10000)
        date_achat = fake.date_between(start_date='-10y',
end_date='today').strftime('%Y-%m-%d')
        # Generate a random price within the specified range, rounded to two decimal
places
        prix = round(random.uniform(*price_range), 2)
```

```
        # Write a row of data in the specified format
        writer.writerow([i+1, id_equipement, name, type_equipement, date_achat, prix])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()

# Generate 10 formatted entries for 'Equipements'
formatted_data_equipements = generate_equipements_formatted(360, equipments_list)
# Adjusted function to generate formatted data for TypeDeCours
def generate_types_de_cours_formatted(n, course_list):
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    for i in range(n):
        id_type_de_cours = fake.unique.random_int(min=1, max=1000)
        type_type_de_cours = course_list[i % len(course_list)]  # Use modulo for
cycling through the list if n > len(course_list)

        # Write a row of data in the specified format
        writer.writerow([i+1, id_type_de_cours, type_type_de_cours])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()

# Generate formatted entries for 'TypeDeCours'
formatted_data_types_de_cours =
generate_types_de_cours_formatted(len(types_de_cours_list), types_de_cours_list)
# Function to generate data for Entraineur
# Define the specialties list as provided
specialites = types_de_cours_list

# Adjusted function to generate formatted data for Entraineurs
def generate_entraineurs_formatted(n, specialites):
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)

    for i in range(n):
        id_entraineur = fake.unique.random_int(min=1, max=10000)
        nom = fake.last_name()
        prenom = fake.first_name()
        specialite = random.choice(specialites)
        date_embauche = fake.date_between(start_date='-10y',
end_date='today').strftime('%Y-%m-%d')

        # Write a row of data in the specified format
        writer.writerow([i+1, id_entraineur, nom, prenom, specialite, date_embauche])

    # Get the string from the string buffer
    output.seek(0)
    return output.read()
```

```python
# Generate 10 formatted entries for 'Entraineurs'
formatted_data_entraineurs = generate_entraineurs_formatted(90, specialites)
def convert_to_dict_ignoring_first(input_str):
    # Split the input string into lines
    lines = input_str.strip().split('\n')

    # Define the list to hold dictionaries
    dict_list = []

    # Define the keys for the dictionaries
    keys = ['ID_Entraineur', 'Nom', 'Prénom', 'Spécialité', 'DateEmbauche']

    # Iterate over each line
    for line in lines:
        # Split the line by comma, strip the quotes, and ignore the first element
        elements = [e.strip('"') for e in line.split(',')[1:]]

        # Create a dictionary for the line and append to the list
        dict_list.append(dict(zip(keys, elements)))

    return dict_list

# Test the function with the provided string
entraineurs_data = convert_to_dict_ignoring_first(formatted_data_entraineurs)
def convert_to_course_dict_corrected(input_str):
    # Split the input string into lines
    lines = input_str.strip().split('\n')

    # Define the list to hold dictionaries
    dict_list = []

    # Define the keys for the dictionaries
    keys = ['ID_TypeDeCours', 'Type_TypeDeCours']

    # Iterate over each line
    for line in lines:
        # Split the line by comma and strip the quotes
        elements = [e.strip('"') for e in line.split(',')]

        # Swap the elements to match the correct key-value pair
        elements = [elements[1], elements[2].strip('"\r')]

        # Create a dictionary for the line and append to the list
        dict_list.append(dict(zip(keys, elements)))

    return dict_list

# Test the corrected function with the provided string
type_de_cours_data = convert_to_course_dict_corrected(formatted_data_types_de_cours)
# Function to generate data for Cours
def generate_cours_formatted(n, entraineurs, types_de_cours):
```

```
    output = StringIO()
    writer = csv.writer(output, quoting=csv.QUOTE_NONNUMERIC)
    # Generate data
    for i in range(n):
        id_cours = fake.unique.random_int(min=1, max=10000)
        entraineur = random.choice(entraineurs)
        type_de_cours = random.choice(types_de_cours)
        nom = f"{type_de_cours['Type_TypeDeCours']} with {entraineur['Prénom']}
{entraineur['Nom']}"
        description = f"{type_de_cours['Type_TypeDeCours']} class provided by
{entraineur['Prénom']} {entraineur['Nom']} specializing in
{entraineur['Spécialité']}."
        # Write the data to CSV writer
        writer.writerow([i+1, id_cours, type_de_cours['ID_TypeDeCours'],
entraineur['ID_Entraineur'], nom, description])
    # Return the generated CSV data
    output.seek(0)
    return output.read()

# Generate the formatted data for 10 Cours entries
formatted_data_cours = generate_cours_formatted(300, entraineurs_data,
type_de_cours_data)
def write_to_file(input_list, filename):
    # Ouvrir le fichier en mode 'write', cela va créer le fichier s'il n'existe pas,
ou le réécrire s'il existe déjà
    with open(filename, 'w') as f:
        # Écrire chaque élément de la liste dans le fichier comme une ligne distincte
        for line in input_list:
            f.write(line + '\n')

input_string = [formatted_data_salle,
                formatted_data_membres,
                formatted_data_moyen_paiement,
                formatted_data_abonnements,
                formatted_data_equipements,
                type_de_cours_data,
                entraineurs_data,
                formatted_data_cours]

output_filename = ['SalleMusculation.txt',
                   'Membres.txt',
                   'MoyenPaiement.txt',
                   'Abonnements.txt',
                   'Equipements.txt',
                   'TypeDeCours.txt',
                   'Entraineurs.txt',
                   'Cours.txt']

for i in range(len(input_string)):
    write_to_file(input_string[i], output_filename[i])
```

Projet de Base de Données IN513
Université de Versailles-Saint-Quentin-en-Yvelines

AMOUSSOU Nathan
JEYAKANTHAN Thushanth
LAM Mathieu

2023-2024

Table des matières

1 Introduction

Pour ce projet en base de données, nous avons conçu un cahier des charges détaillé visant à élaborer une base de données pour la gestion des salles de musculation. Cette initiative s'inscrit dans un contexte où la musculation et le fitness ont pris une place prépondérante dans notre société contemporaine, non seulement comme activités de loisir mais également comme éléments essentiels à une vie saine et équilibrée. Les bienfaits du sport, et plus spécifiquement de la musculation, sont multiples et reconnus. Ils contribuent de manière significative à l'amélioration de la condition physique, au renforcement musculaire, à la perte de poids, au maintien de l'équilibre psychologique et au renforcement de la confiance en soi.

Avec la démocratisation de l'accès au sport, les salles de musculation sont devenues des espaces inclusifs, accueillant une diversité de profils, des athlètes chevronnés aux débutants, en passant par ceux qui cherchent à reprendre une activité physique régulière après une période d'inactivité. Elles offrent une gamme variée d'abonnements, rendant le fitness plus accessible à tous les niveaux de revenus. Les équipements modernes et les différentes gammes de cours proposés sont conçus pour répondre à un large éventail de besoins et de préférences, permettant à chaque individu de trouver une activité adaptée à ses objectifs personnels.

En tant que membres actifs de salles de musculation, nous sommes témoins de l'évolution constante de ces espaces, qui s'adaptent sans cesse aux dernières tendances et innovations en matière de fitness. Notre intérêt personnel pour le sujet nous a poussé à le choisir comme support pour ce projet d'IN513, où nous chercherons à comprendre et à modéliser les complexités inhérentes à la gestion d'une salle de sport. Notre projet aspire à simuler un environnement réaliste de gestion, reflétant les défis quotidiens tels que le suivi des membres, la gestion des abonnements, l'organisation des cours et l'entretien des équipements.

À travers ce projet, nous avons l'opportunité d'appliquer nos connaissances théoriques dans un cadre pratique, en construisant un système de gestion de base de données qui non seulement répond aux exigences fonctionnelles d'une salle de musculation mais qui favorise également une meilleure compréhension des besoins et des comportements de ses utilisateurs. Notre objectif est de créer une base de données robuste, flexible et évolutive, capable de s'adapter aux évolutions futures du secteur du fitness, tout en offrant une expérience utilisateur optimisée pour les gestionnaires et les membres des salles de musculation.

2 Modèle Entité-Association

Le schéma Entité-Association capture l'essence des données et leurs connexions, fournissant une représentation graphique qui aide à structurer la base de données. C'est une étape déterminante pour un design qui reflète fidèlement les opérations de nos salles de musculation.

2.1 Schéma

Le schéma Entité-Association est un composant clé dans la conception de bases de données, illustrant de manière visuelle les entités, leurs attributs, et les liens qui les unissent. Cet outil de modélisation facilite la compréhension des structures de données et guide la création d'un système organisé et efficace.

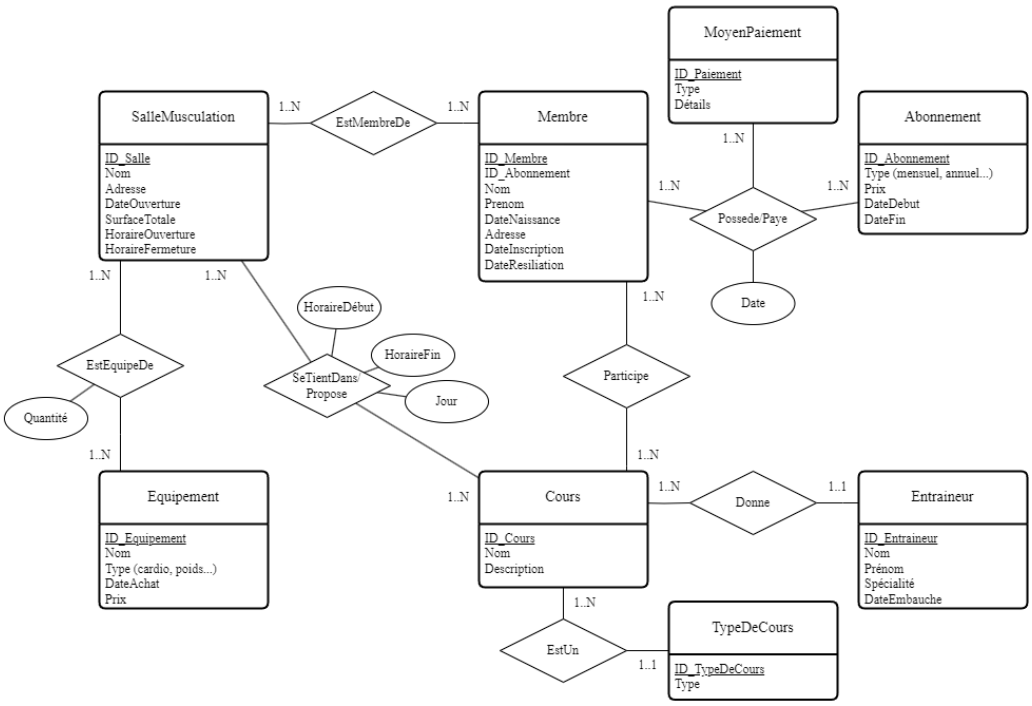

FIGURE 1 – Schéma du modèle Entité-Association

2.2 Entités et attributs

Dans la modélisation de notre problème, nous avons conceptualisé sept entités, afin de pouvoir les construire dans la deuxième partie du projet.

- **Salle de Musculation** : Chaque salle de musculation possède un identifiant unique, un nom, une adresse, une date d'ouverture, une superficie totale, ainsi que des horaires d'ouverture et de fermeture.
- **Membre** : Chaque membre est identifié par un numéro unique, est lié à un abonnement, et a un nom, un prénom, une date de naissance, une adresse, une date d'inscription et éventuellement une date de résiliation.
- **Abonnement** : Chaque abonnement a un identifiant unique, un type, un prix, une date de début et potentiellement une date de fin.
- **Équipement** : Chaque équipement est identifié par un numéro unique, a un nom, un type, une date d'achat et un prix.
- **Cours** : Chaque cours a un identifiant unique, a un nom et une description
- **TypeDeCours** : Chaque type de cours possède un identifiant unique et un type
- **Entraîneur** : Chaque entraîneur a un identifiant unique, un nom, un prénom, une spécialité, et une date d'embauche.
- **Moyen de Paiement** : Chaque moyen de paiement a un identifiant unique, un type et des des détails qui varient en fonction du type

2.3 Relations

Dans notre modèle de base de données des salles de musculation, nous pouvons identifié plusieurs relations entre les entités. Ces relations sont essentielles dans la gestion des données de la salle étant donné que celles-ci servent à modéliser la manière dont les entités sont liées les unes aux autres.

- Les membres peuvent s'inscrire dans plusieurs salles de musculation, et chaque salle peut avoir de nombreux membres.
- Un membre est lié à un seul abonnement, mais un abonnement peut concerner plusieurs membres.
- Les salles de musculation sont équipées de plusieurs équipements et un équipement peut se trouver dans plusieurs salles.
- Un entraîneur peut donner plusieurs cours, mais un cours est enseigné par un seul entraîneur.
- Les membres peuvent participer à plusieurs cours, et chaque cours peut accueillir de nombreux membres.
- Un cours ne peut être que d'un seul type, celui-ci se déroule dans une seule salle de musculation, bien que cette salle puisse accueillir plusieurs cours.
- Un membre peut utiliser plusieurs moyens de paiement, mais un moyen de paiement est associé à un seul membre.
- Un moyen de paiement peut être utilisé pour plusieurs abonnements, mais un abonnement est payé par un seul moyen de paiement.

2.4 Contraintes d'intégrité

Une base de données sans contraintes d'intégrité serait chaotique et peu fiable. Les contraintes assurent la validité et la fiabilité des données en définissant les règles que les données doivent respecter pour être considérées comme valides. Cette sous-section va donc détailler les contraintes d'intégrité de notre modèle, garantissant que les données restent précises, cohérentes et conformes à nos attentes et besoins opérationnels.

- Les salles de musculation doivent avoir une adresse unique et (doivent déjà être ouverte) ne peuvent pas être ouvertes dans le futur. Leur superficie doit être positive et les horaires d'ouverture doivent précéder ceux de fermeture.
- Les membres doivent être déjà inscrits, et leur date de naissance doit également précédée la date d'inscription. L'inscription à un abonnement doit correspondre à la date de début de cet abonnement, et la résiliation à la date de fin.
- Les abonnements doivent avoir un prix positif ou nul et les dates de début et de fin doivent correspondre respectivement à l'inscription et la résiliation du membre associé.
- L'achat d'équipement doit être déjà fait et le prix doit être positif ou nul.
- Les cours doivent être planifiés pour le futur et les horaires doivent être cohérents, avec un début avant la fin.
- Les entraîneurs doivent être préalablement embauchés et avoir une spécialité déclarée.
- Les moyens de paiement doivent être déjà ajoutés et doivent être liés à un membre spécifique. Les détails du moyen de paiement doivent être traités avec précaution en raison de leur nature potentiellement sensible.
- Dans la table SalleMusculation, le numéro de ID_Salle doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1.
- Dans la table Membre, le numéro de ID_Membre doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1
- Dans la table Abonnement, le numéro de ID_Abonnement doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1
- Dans la table Equipement, le numéro de ID_Equipement doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1
- Dans la table Cours, le numéro de ID_Cours doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1
- Dans la table Entraineur, le numéro de ID_Entraineur doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1
- Dans la table Entrainement, le numéro de ID_Paiement doit être égal à celui de la commande précédente + 1 et la première commande porte le numéro 1
- Pour tous les abonnements dont la date de début (DateDebut) de l'abonnement est comprise pendant la période du 20 décembre au 10 janvier, les frais d'adhésion/inscription sont offerts (il faut ajouter l'élément frais d'adhésion)
- Pour tous les abonnements, le premier mois est offert
- Empêcher la suppression d'un membre dont l'abonnement est toujours en cours
- Un abonnement doit être lié à un membre existant
- L'âge d'un membre doit être supérieur à 18
- Un membre doit avoir un abonnement actif avant de pouvoir réserver un cours

3 Modèle relationnel

Le modèle relationnel est au cœur de la conception de notre système de base de données. Il articule comment les données sont structurées et comment les différentes entités interagissent entre elles. Cette section détaillera le modèle relationnel de notre base de données pour la gestion des salles de musculation, en mettant l'accent sur les entités, les attributs et les relations qui permettront de stocker et de récupérer des informations de manière efficace et cohérente.

3.1 Entités attributs

Les fondations d'une base de données robuste sont posées à travers la définition précise des entités et de leurs attributs. Cette sous-section met en lumière la structure de chaque entité en détail, en énumérant leurs caractéristiques distinctives et la nature des données qu'elles conserveront. Chaque entité est pensée pour refléter un aspect spécifique et nécessaire de l'écosystème de notre salle de musculation.

- **Salle de Musculation** : Chaque salle de musculation possède un numéro d'identification unique, un nom, une adresse spécifique, une date à laquelle elle a commencé à opérer, une mesure de l'espace disponible pour l'entraînement, et les heures auxquelles elle ouvre et ferme chaque jour.
- **Membre** : Chaque personne qui s'inscrit à la salle de musculation reçoit un numéro d'identification unique, est rattachée à un abonnement spécifique, et doit fournir son nom, son prénom, sa date de naissance, son adresse personnelle, la date à laquelle elle s'est inscrite, et éventuellement la date à laquelle elle a mis fin à son abonnement.
- **Abonnement** : Les abonnements sont caractérisés par un numéro d'identification unique, un type qui décrit la durée de l'abonnement (comme mensuel ou annuel), un prix, une date de début de validité, et une date de fin éventuelle.
- **Équipement** : Chaque pièce d'équipement dans la salle de musculation a un numéro d'identification unique, un nom, une catégorie décrivant son usage (comme cardio ou musculation), une date d'acquisition, et un prix.
- **Cours** : Les cours offerts par la salle de musculation sont identifiés par un numéro unique, et ont un nom, un type décrivant l'activité (comme le yoga ou le spinning).
- **TypeDeCours** : Les types de cours incarnés par les cours sont identifiés par un numéro unique, ils possèdent un type (yoga, spinning, cardio, cross, musculation, biking ...)
- **Entraîneur** : Chaque entraîneur a un numéro d'identification unique, un nom, un prénom, une spécialité dans un domaine particulier de l'entraînement, et une date à laquelle il ou elle a commencé à travailler dans la salle.
- **Moyen de Paiement** : Les moyens de paiement sont identifiés par un numéro unique et sont caractérisés par un type (comme carte de crédit ou PayPal) et des détails spécifiques qui varient en fonction du type de paiement.

3.2 Relations

Les relations entre les entités sont le fil conducteur qui permet de naviguer dans le labyrinthe des données et d'en extraire du sens. Ici, nous allons définir les différents types de liens qui existent entre nos entités, en expliquant comment elles interagissent les unes avec les autres. Ces relations sont essentielles pour modéliser la complexité des opérations dans les salles de musculation et pour permettre des requêtes multifacettes sur la base de données.

- **EstMembreDe** : Cette relation indique quel membre est inscrit dans une salle de musculation donnée.
- **EstÉquipéDe** : Cette relation répertorie le nombre d'équipements présents dans une salle de musculation donnée.
- **SeTientDans/Propose** ; Cette relation répertorie dans quelle salle de musculation un cours donné se tient ou bien quelles sont les cours proposés dans une salle de musculation donnée.
- **EstUn** : Cette relation indique de quel type est un cours donné.
- **Donne** : Elle indique quel entraîneur donne un cours donné.
- **Participe** : Cette relation révèle quels membres participent à un cours donnés.
- **Possede/Paye** : Elle montre l'abonnement qui est possédé par un membre donné ou bien quels moyens de paiement sont utilisés pa membre donné.

3.3 Contraintes d'intégrité

Une base de données sans contraintes d'intégrité serait chaotique et peu fiable. Les contraintes assurent la validité et la fiabilité des données en définissant les règles que les données doivent respecter pour être considérées comme valides. Cette sous-section va donc détailler les contraintes d'intégrité de notre modèle, garantissant que les données restent précises, cohérentes et conformes à nos attentes et besoins opérationnels.

- **Salle de Musculation** : Un identifiant unique pour chaque salle, une adresse non redondante, une ouverture préalable, des dimensions positives, et des heures d'ouverture et de fermeture logiques.
- **Membre** : Un identifiant unique par membre, association avec un abonnement valide, informations personnelles complètes, inscription préalable, et correspondance des dates de résiliation avec l'abonnement.
- **Abonnement** : Un identifiant unique, un type défini, un prix non négatif, et des dates de début et de fin correspondant aux dates d'inscription et de résiliation des membres.
- **Équipement** : Un identifiant unique, une désignation claire, une catégorie spécifique, un achat dans le passé, et un prix non négatif.
- **Cours** : Un identifiant unique, une association à une salle spécifique, une désignation et un type clairs, programmés pour le futur, avec des horaires cohérents.
- **Entraîneur** : Un identifiant unique, des informations personnelles complètes, une spécialité définie, et un engagement dans le passé.
- **Moyen de Paiement** : Un identifiant unique, un type défini avec des détails appropriés, une association avec un membre spécifique, et une mise en place dans le passé.

4 Rôles

Dans la conception de notre base de données, il est primordial de définir clairement les différents rôles et leurs permissions respectives. Ces rôles sont essentiels pour assurer la sécurité des données et pour que chaque utilisateur interagisse avec le système de manière appropriée, en fonction de ses responsabilités. Nous avons identifié quatre rôles principaux, chacun avec des niveaux d'accès et de

contrôle spécifiques, garantissant ainsi une gestion efficace et ordonnée des opérations quotidiennes de la salle de musculation.

- **Administrateur** : Accès complet pour ajouter, modifier, supprimer et lire toutes les données. Gestion des utilisateurs et des rôles inclus.
- **GestionnaireDeSalle** : Gestion des informations relatives à une salle spécifique, y compris les équipements, les cours, et les paiements.
- **Entraineur** : Permissions pour ajouter, modifier et supprimer les cours qu'il donne.
- **Membre** : Accès en lecture seule à ses informations personnelles, cours, moyens de paiement, et transactions avec la possibilité de mise à jour.

5 Questions à poser sur le modèle

La conception d'un modèle de base de données exige une réflexion approfondie sur les questions auxquelles le système devra répondre. Ces questions guident la structuration des données et les relations entre elles. Elles servent également à vérifier la pertinence du modèle et sa capacité à fournir des informations utiles pour la prise de décision. Voici une série de questions potentielles que notre modèle de base de données devrait être en mesure d'aborder pour illustrer la diversité et la profondeur des insights que nous souhaitons extraire de notre système.

- Quel est le pourcentage d'utilisation des équipements de cardio par rapport aux équipements de musculation dans l'ensemble de nos salles ?
- Quelle salle de musculation a le taux de renouvellement d'abonnement le plus élevé et quelles pourraient en être les raisons (basées sur les types de cours, les équipements disponibles, etc.) ?
- Quel est le profil démographique (âge, sexe) des membres qui fréquentent le plus les cours de haute intensité comme le CrossFit ou le HIIT ?
- Comment les habitudes de paiement ont-elles évolué au cours des dernières années (carte de crédit, prélèvement automatique, PayPal, etc.) et quelles stratégies pourrions-nous adopter pour optimiser les processus de paiement ?
- Quels sont les entraîneurs qui ont le plus grand nombre de membres inscrits à leurs cours et comment cela se compare-t-il à leur ancienneté et à leurs spécialités ?
- Pouvez-vous identifier les périodes de l'année où les inscriptions aux salles de musculation sont les plus élevées et les corréler avec les campagnes marketing ou les périodes de promotions spéciales ?
- Quelle corrélation existe-t-il entre la superficie d'une salle de musculation et le nombre d'équipements qu'elle contient ? Cette densité a-t-elle un impact sur la fréquentation des membres ?
- Est-il possible de prévoir la durée de vie d'un équipement en fonction de son prix et de sa fréquence d'utilisation, afin de mieux planifier les achats futurs ?
- Quelle est la durée moyenne entre l'inscription d'un membre et sa première participation à un cours, et quels facteurs pourraient accélérer cette intégration ?
- Peut-on déterminer un modèle de comportement des membres résiliant leur abonnement (fréquence des visites, types de cours suivis, durée depuis l'inscription, etc.) pour développer des stratégies de rétention ?