Project M1

Advanced Data Bases

M1 – Apprentissage - Ingénieur Big Data et Machine Learning/IA

EFREI Paris

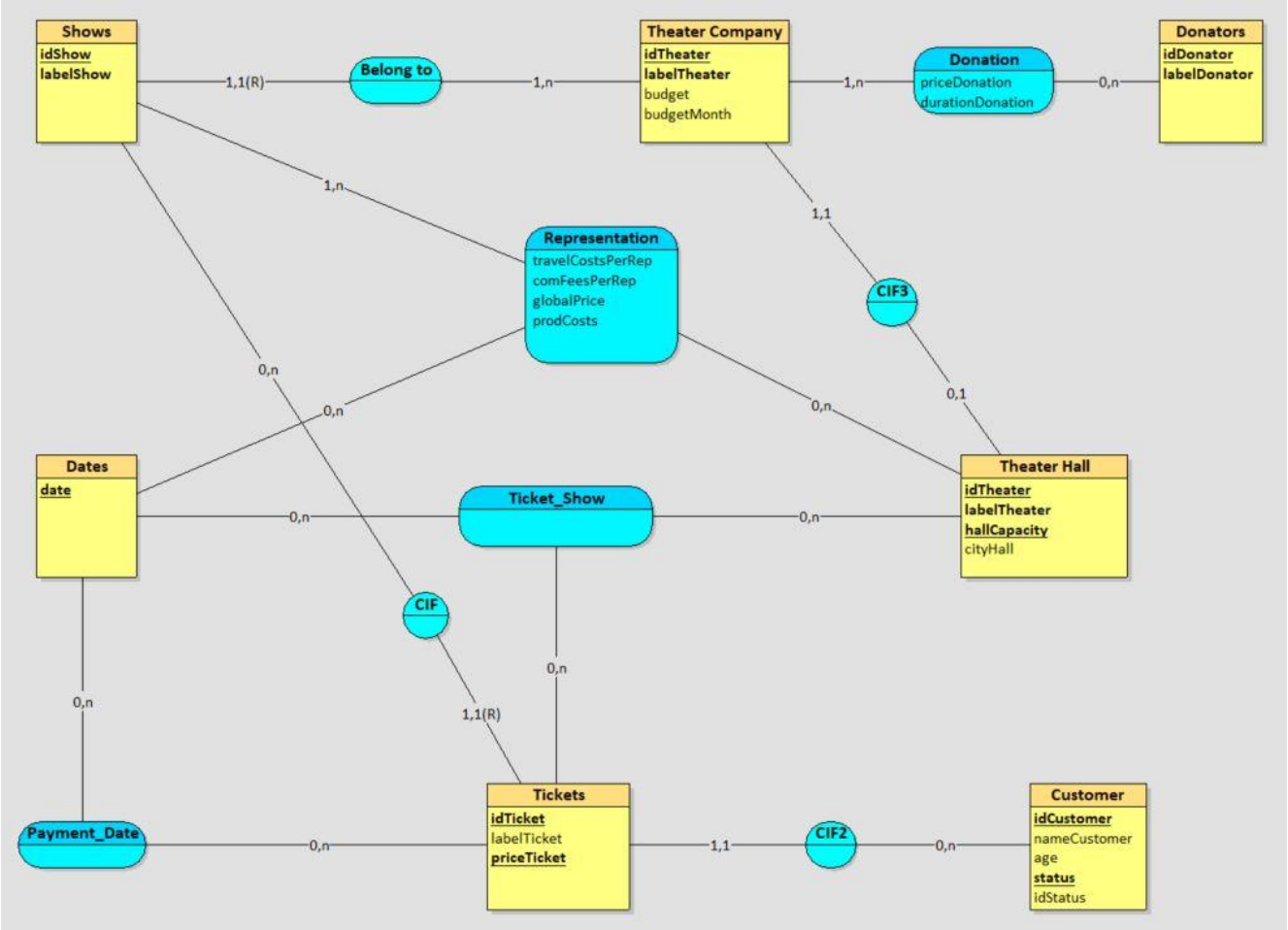
Arthur ALLIÉ

Elysé RASOLOARIVONY

30 - 01 - 2022

I. MCD

Ce MCD comporte 7 tables différentes et 4 tables associatives, faisant la jonction entre ces différentes tables.



Les 7 tables sont les suivantes :

- **DATES**

Cette table fournit la date de paiement à la table TICKETS ainsi que les dates de représentations des différents spectacles aux tables associatives TICKET_SHOW et REPRESENTATION.

- **DONATORS**

Cette table liste tous les différents contributeurs de la base de données, qui subventionnent les compagnies de théâtre de leur choix avec des dons mensuels étalés sur une certaine période.

- **CUSTOMER**

Cette table répertorie tous les clients de spectacles de la base de données. Il y a 5 types de statuts de clients possibles : normal, elderly, unemployed, student ou child (champ status).

- **THEATER_HALL**

Cette table répertorie toutes les salles de théâtre de la base de données ; ces salles possèdent entre autres un nom (labelTheater) ainsi qu'une capacité d'accueil (hallCapacity). La clé primaire de cette table est composée du couple (idTheater, hallCapacity). En effet il est notamment utile de récupérer hallCapacity dans la table associative TICKET_SHOW, pour gérer une condition d'insertion en fonction du taux de remplissage d'une salle de spectacle.

- **THEATER_COMPANY**

Cette table possède comme clé primaire le champ idTheater, qui est aussi clé étrangère de la table THEATER_HALL (voir ci-dessus et partie MLD pour une meilleure compréhension). Son champ labelTheater est identique à celui de la table THEATER_HALL ; en effet une compagnie de théâtre possède ici le nom de la salle à laquelle elle est rattachée. Exemple : La compagnie de théâtre 'Temple Solaire' est rattachée à la salle de théâtre 'Temple Solaire'.

- **SHOWS**

Cette table possède comme clé primaire le couple (idShow, idTheater), avec idTheater la clé primaire de la compagnie de théâtre à laquelle est rattaché le spectacle considéré. L'identifiant relatif 1,1(R) présent entre la table SHOWS et la table THEATER_COMPANY permet conceptuellement de numéroter successivement les spectacles rattachés à une compagnie de théâtre donnée à partir de 1, puis de recommencer la numérotation à partir de 1 quand on passe à la compagnie de théâtre suivante, etc. Ainsi, idTheater est aussi clé étrangère en référence à idTheater de la table THEATER_COMPANY.

- **TICKETS**

Cette table enregistre chacun des tickets achetés par les différents clients dans la base de données. Chacun de ces tickets possède un type particulier (champ labelTicket) parmi 5 possibilités, correspondant aux 5 statuts de clients possibles (voir champ 'statusCustomer' et aussi table CUSTOMER ci-dessus) :

- Client au statut normal => champ labelTicket = normalPriceTicket & champ priceTicket = 25
- Client au statut elderly => champ labelTicket = reducedPriceTicket & champ priceTicket = 20
- Client au statut unemployed => champ labelTicket = reducedPriceTicket & champ priceTicket = 15
- Client au statut student => champ labelTicket = reducedPriceTicket & champ priceTicket = 15
- Client au statut child => champ labelTicket = reducedPriceTicket & champ priceTicket = 10

Cette table possède en sus (idTicket, priceTicket, idShow, idTheater) comme clé primaire ; le champ priceTicket permet en effet la gestion des opérations d'insertion ultérieures dans la table TICKET_SHOW (voir ci-dessous).

Cette table récupère donc le couple (idShow, idTheater) dans sa clé primaire. Ce couple provient de la table SHOWS et est identifiant relatif de la table TICKETS. Ainsi, ce couple est aussi clé étrangère en référence au couple (idShow, idTheater) de la table SHOWS.

<u>Les 4 tables associatives sont comme suit :</u>

- **DONATION**

Cette table associative entre THEATER_COMPANY et DONATORS possède deux champs qui lui sont propres :

- priceDonation, qui représente le montant mensuel du don du contributeur, qui subventionne la compagnie de théâtre de son choix.

- durationDonation, qui représente la durée de versement de ce montant (en mois).

- **PAYMENT_DATE**

Cette table associative entre DATES et TICKETS enregistre les différents paiements de chacun des tickets, en leur associant leur date de paiement respective (champ dat). Les contraintes d'intégrité fonctionnelle du MCD font que la clé primaire de cette table est composée des champs (dat, idTicket, priceTicket, idShow, idTheater). En effet, chaque ligne de cette table indique la date d'achat d'un ticket, ce ticket étant rattaché à un spectacle particulier, lui-même rattaché à une compagnie de théâtre particulière (voir MCD ci-dessus).

- **REPRESENTATION**

Cette table associative se trouve entre les tables DATES, THEATER_HALL et SHOWS. Elle répertorie chacune des différentes représentations des spectacles données, chaque spectacle ayant une date particulière qui lui est associée (champ dat provenant de la table DATES) ainsi qu'une salle de spectacle dans laquelle la représentation a lieu (représenté par le couple idTheaterHall, hallCapacity). Naturellement, chaque représentation d'un spectacle possède un couple (idShow, idTheaterCompany), représentant respectivement le spectacle et la compagnie de théâtre à laquelle appartient ce spectacle. Ainsi, elle possède donc pour clé primaire (dat, idShow, idTheaterCompany, idTheaterHall, hallCapacity).

De plus, cette table possède les champs suivants :

- travelCostsPerRep : il s'agit des coûts de transports engagés par le théâtre qui produit lorsque le spectacle est donné dans une salle extérieure à la salle de la compagnie de théâtre productrice du spectacle

- comFeesPerRep : il s'agit des honoraires dus aux comédiens du spectacle chaque soir de chaque représentation

- globalPrice : il s'agit du montant encaissé par le théâtre propriétaire du spectacle auprès du théâtre accueillant ce spectacle dans sa propre salle. Ce montant est payé en une fois le jour de la première représentation du spectacle dans la salle de théâtre extérieure correspondante. Donc, pour des dates se suivant dans la table REPRESENTATION avec un (idShow, idTheaterCompany, idTheaterHall) particulier, ce montant sera enregistré une seule fois seulement.

- prodCosts : il s'agit des coûts de production (staging costs), engagés une seule fois le premier soir de chaque spectacle par la compagnie de théâtre productrice de ce spectacle (de manière analogue à la recette globalPrice, voir ci-dessus). Ces frais comprennent tous les frais de mise en scène exceptés les honoraires des comédiens, c'est-à-dire : l'éclairage, les décors, les costumes, etc.

Cette table correspond uniquement aux valeurs d'insert du théâtre propriétaire du spectacle.

Les insert du théâtre accueillant la représentation d'un spectacle qui n'est pas le sien sont gérés par le trigger majBudgTheaterRep (voir plus bas).

- **TICKET_SHOW**

Cette table associative se trouve entre les tables DATES, THEATER_HALL et TICKETS. Cette table répertorie les différents tickets achetés par des clients, elle s'appuie donc sur la table TICKETS dont elle récupère les champs (idTicket, priceTicket, idShow, idTheaterCompany). Elle récupère aussi le champ dat provenant de la table DATES, qui permet d'associer à un ticket une date de représentation du spectacle auquel il est associé. Enfin, elle récupère le couple idTheaterHall, hallCapacity de la table THEATER_HALL (voir le MCD). Comme évoqué plus haut, il est utile de récupérer hallCapacity dans la table associative TICKET_SHOW, pour gérer une condition d'insertion en fonction du taux de remplissage d'une salle de spectacle (voir le trigger insTicketsShowRepDate plus bas). TICKET_SHOW a donc comme clé primaire l'ensemble de champs suivants :

(dat, idTicket, priceTicket, idShow, idTheaterCompany, idTheaterHall, hallCapacity)

Remarque : compte tenu de la longueur de certaines tables de cette base de données (fichier *insertion.sql*), il est conseillé à l'utilisateur de ne pas toutes les copier en une seule fois dans LiveSQL.

II. MLD

- **DATES (dat)**
  dat : clé primaire

- **DONATORS (idDonator, labelDonator)**
  idDonator : clé primaire

- **CUSTOMER (idCustomer, nameCustomer, age, status, idStatus)**
  idCustomer, status : clé primaire

- **THEATER_HALL (idTheater, labelTheater, hallCapacity, cityHall)**
  idTheater, hallCapacity : clé primaire

- **THEATER_COMPANY (*#idTheater*, labelTheater, *#hallCapacity*, budget, budgetMonth)**
  idTheater : clé primaire
  idTheater, hallCapacity : clé étrangère en référence à idTheater, hallCapacity de THEATER_HALL

- **SHOWS (idShow, labelShow, #*idTheater*)**
  idShow, idTheater : clé primaire
  idTheater : clé étrangère en référence à idTheater de THEATER_COMPANY

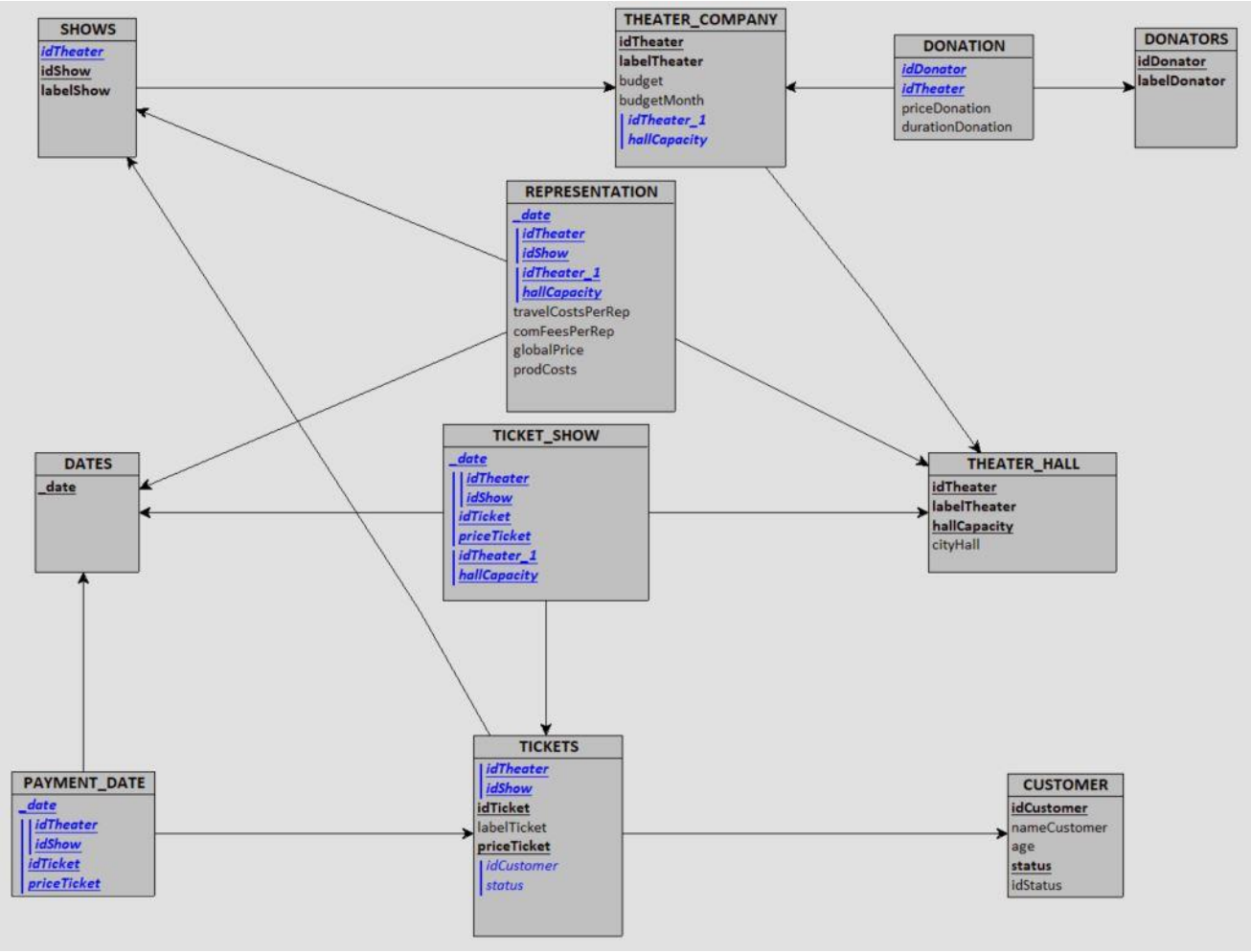
- **TICKETS (idTicket, labelTicket, priceTicket, #*idShow*, #*idTheater*, #*idCustomer, #statusCustomer*)**
  idTicket, priceTicket, idShow, idTheater : clé primaire
  idShow, idTheater : clé étrangère en référence à idShow, idTheater de SHOWS
  idCustomer, statusCustomer : clé étrangère en référence à idCustomer, status de CUSTOMER

- **DONATION (#*idDonator*, #*idTheater*, priceDonation, durationDonation)**
  idDonator, idTheater : clé primaire
  idDonator : clé étrangère en référence à idDonator de DONATORS
  idTheater : clé étrangère en référence à idTheater de THEATER_COMPANY

- **PAYMENT_DATE (#*dat*, #*idTicket*, #*priceTicket*, #*idShow*, #*idTheater*)**
  dat, idTicket, priceTicket, idShow, idTheater : clé primaire
  dat : clé étrangère en référence à dat de DATES
  idTicket, priceTicket, idShow, idTheater : clé étrangère en référence à idTicket, priceTicket, idShow, idTheater de TICKETS

- **REPRESENTATION (#*dat*, #*idShow*, #*idTheaterCompany*, #*idTheaterHall*, #*hallCapacity*, travelCostsPerRep, comFeesPerRep, globalPrice, prodCosts)**
  dat, idShow, idTheaterCompany, idTheaterHall, hallCapacity : clé primaire
  dat : clé étrangère en référence à dat de DATES
  idShow, idTheaterCompany : clé étrangère en référence à idShow, idTheater de SHOWS
  idTheaterHall, hallCapacity : clé étrangère en référence à idTheater, hallCapacity de THEATER_HALL

- **TICKET_SHOW (#*dat*, #*idTicket*, #*priceTicket*, #*idShow*, #*idTheaterCompany*, idTheaterHall, hallCapacity)**
  dat, idTicket, priceTicket, idShow, idTheaterCompany, idTheaterHall, hallCapacity : clé primaire
  dat : clé étrangère en référence à dat de DATES
  idTicket, priceTicket, idShow, idTheaterCompany : clé étrangère en référence à idTicket, priceTicket, idShow, idTheater de TICKETS idTheaterHall, hallCapacity : clé étrangère en référence à idTheater, hallCapacity de THEATER_HALL



III. TRIGGERS D'INSERTION EN BASE DE DONNÉES

Triggers principaux (fin du fichier insertion.sql)

- **TRIGGER insCustomerTicket**

Ce trigger permet d'attribuer un prix de billet de base pour chacune des différentes catégories de clients (normal, elderly, unemployed, student et child), avant l'insertion d'une ligne dans la table TICKET.

- **TRIGGER insTicketsShowRepDate**

Ce trigger permet la modification du prix de base des tickets en fonction de la date d'achat de ceux-ci et du taux de remplissage de la salle.

- **TRIGGER insertExpenditure**

Ce trigger sert à s'assurer que les insert dans la table REPRESENTATION ont des signes arithmétiques cohérents avec la situation de spectacle concernée : une compagnie qui accueille son propre spectacle ou qui l'exporte dans une autre salle n'aura pas les mêmes dépenses ni les mêmes recettes (globalPrice) dans l'un ou l'autre des deux cas.

- **TRIGGER majBudgTheaterRep**

Ce trigger sert à mettre à jour, dans la table THEATER_COMPANY, le champ budget du théâtre accueillant la représentation d'un spectacle, après chaque insert dans la table REPRESENTATION.

- **TRIGGER majBudgTheaterDonation**

Ce trigger sert à alimenter la table THEATER_COMPANY après chacun des dons, qui ont lieu chaque mois.

- **TRIGGER majBudgTheaterTickets**

Ce trigger sert à alimenter la table THEATER_COMPANY au début de chaque mois avec la billetterie du mois précédent, intégrée dans le budget du théâtre concerné.

Triggers supplémentaires (fin du fichier insertion.sql)

- **TRIGGER insertionTicket**

Ce trigger sert à vérifier que le client auquel est rattaché le ticket existe en BDD dans la table CUSTOMER. Cela peut être utile dans le cas de l'insert à la main de la part d'un compte administrateur.

- **TRIGGER insPaymentDate**

Ce trigger sert à insérer une ligne dans la table PAYMENT_DATE après chaque insert dans la table TICKETS.

- **TRIGGER insertGoodShowId**

Ce trigger sert à vérifier l'existence dans la table SHOWS d'un numéro de spectacle en rapport avec l'ajout d'une nouvelle ligne dans la table REPRESENTATION.

- **TRIGGER insertGoodHallID**

Ce trigger sert à vérifier que la nouvelle ligne dans la table REPRESENTATION est rattachée à une salle de spectacle connue dans la table THEATER_HALL.

IV. PROCÉDURES STOCKÉES (fichier requetes.sql)

ORGANIZATION

- **TRIGGER insertionRepresentation**

Ce trigger permet de vérifier que le théâtre qui accueille la représentation à une date donnée n'a pas déjà d'autre représentation à cette date.

- **TRIGGER insertionRepresentation2**

Ce trigger sert à vérifier que la compagnie n'a pas déjà programmée un spectacle dans une autre salle.

- **PROCEDURE setOfCities**

Cette procédure sert à afficher les différentes villes dans lesquelles joue une compagnie de théâtre durant une période de temps donnée.

```
1
2  drop procedure setOfCities;
3
4  CREATE OR REPLACE PROCEDURE
5  setOfCities (nametheater VARCHAR2, dateshow1 DATE, dateshow2 DATE) IS
6  BEGIN
7      DECLARE
8          CURSOR cities IS
9              SELECT
10                 tc.labelTheater
11                 ,th.cityHall
12                 ,r.dat
13                 ,s.labelShow
14             FROM REPRESENTATION r
15             JOIN THEATER_HALL th ON r.idTheaterHall = th.idTheater
16             JOIN SHOWS s ON r.idShow = s.idShow AND r.idTheaterCompany = s.idTheater
17             JOIN THEATER_COMPANY tc ON s.idTheater = tc.idTheater
18             WHERE tc.labelTheater = nametheater
19             AND r.dat BETWEEN dateshow1 AND dateshow2
20             GROUP BY tc.labelTheater, th.cityHall, r.dat, s.labelShow;
21
22     BEGIN
23         FOR city IN cities LOOP
24             DBMS_OUTPUT.PUT_LINE('La ville où joue la compagnie ' || city.labelTheater || ' à la date ' || city.dat
25                 || ' est : ' || city.cityHall || '. Le spectacle joué est : ' || city.labelShow);
26             EXIT when cities%NOTFOUND;
27         END LOOP;
28     END;
29
30 END;
31 /
32
33
34 BEGIN
35     setOfCities('Temple Solaire', '29-02-2012', '17-04-2021');
36 END;
37
```

```
Statement processed.
La ville où joue la compagnie Temple Solaire à la date 07-07-2013 est : Daegu. Le spectacle joué est : Tonton était bon
La ville où joue la compagnie Temple Solaire à la date 17-12-2018 est : Valence. Le spectacle joué est : 300
La ville où joue la compagnie Temple Solaire à la date 04-02-2014 est : Alès. Le spectacle joué est : Auguste c'est le minimum
La ville où joue la compagnie Temple Solaire à la date 11-01-2020 est : Alès. Le spectacle joué est : 300
```

TICKETING

- **PROCEDURE ticketPriceToday**

Cette procédure sert à afficher les tarifs des prix des tickets vendus du jour.

```
1
2   drop procedure ticketPriceToday;
3
4   CREATE OR REPLACE PROCEDURE
5   ticketPriceToday(dateshow DATE) IS
6   BEGIN
7       DECLARE
8           CURSOR tickets IS
9               SELECT priceTicket
10              FROM TICKET_SHOW
11              WHERE dat = dateshow
12              ORDER BY priceTicket DESC;
13      BEGIN
14          DBMS_OUTPUT.PUT_LINE('Les prix des tickets du jour sont : ');
15          FOR ticket IN tickets LOOP
16              DBMS_OUTPUT.PUT_LINE(ticket.priceTicket);
17              EXIT when tickets%NOTFOUND;
18          END LOOP;
19      END;
20  END;
21  /
22
23
24  BEGIN
25      ticketPriceToday('30-08-2021');
26  END;
27
```

```
Statement processed.
Les prix des tickets du jour sont :
25
25
20
```

- **PROCEDURE ticketPriceDistribution**

Cette procédure sert à afficher la distribution des tickets des différentes représentations par prix.

```
drop procedure ticketPriceDistribution;

CREATE OR REPLACE PROCEDURE
ticketPriceDistribution IS
BEGIN
    DECLARE
        CURSOR tickets IS
            SELECT
                s.labelShow
                ,tc.labelTheater
                ,ts.dat
                ,ts.priceTicket
                ,COUNT(ts.idTicket) AS distrib
            FROM TICKET_SHOW ts
            JOIN TICKETS t ON ts.idTicket = t.idTicket AND ts.idShow = t.idShow AND ts.idTheaterCompany = t.idTheater
            JOIN SHOWS s ON t.idShow = s.idShow AND t.idTheater = s.idTheater
            JOIN THEATER_COMPANY tc ON s.idTheater = tc.idTheater
            GROUP BY s.labelShow, tc.labelTheater, ts.dat, ts.priceTicket
            ORDER BY (s.labelShow) DESC, ts.priceTicket DESC;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('La distribution par prix des tickets des différentes représentations est la suivante : ');
        FOR ticket IN tickets LOOP
            DBMS_OUTPUT.PUT_LINE(ticket.labelShow || ', ' || ticket.labelTheater || ', à la date du ' || ticket.dat || ', au prix de ' || ticket.priceTicket ||
            ' : ' || ticket.distrib || ' places vendue(s) ');
            EXIT WHEN tickets%NOTFOUND;
        END LOOP;
    END;
END;
/

BEGIN
    ticketPriceDistribution;
END;
```

```
Statement processed.
La distribution par prix des tickets des différentes représentations est la suivante :
Tonton était bon, Temple Solaire, à la date du 07-07-2013, au prix de 25 : 1 places vendue(s)
Tonton était bon, Temple Solaire, à la date du 09-09-2021, au prix de 25 : 1 places vendue(s)
Tonton était bon, Temple Solaire, à la date du 07-07-2013, au prix de 15 : 4 places vendue(s)
Tonton était bon, Temple Solaire, à la date du 09-09-2021, au prix de 10 : 2 places vendue(s)
Les yamnayas, un voyage à travers le temps, Sapien Institute, à la date du 23-05-2017, au prix de 25 : 3 places vendue(s)
Les yamnayas, un voyage à travers le temps, Sapien Institute, à la date du 19-03-2015, au prix de 20 : 5 places vendue(s)
Les yamnayas, un voyage à travers le temps, Sapien Institute, à la date du 23-05-2017, au prix de 20 : 3 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 18-03-2016, au prix de 25 : 2 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 06-06-2013, au prix de 20 : 2 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 18-03-2016, au prix de 20 : 4 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 18-03-2016, au prix de 15 : 1 places vendue(s)
```

- **PROCEDURE labelPriceDistribution**

Cette procédure sert à afficher la distribution des tickets des différentes représentations par type de tarif.

```
CREATE OR REPLACE PROCEDURE
labelPriceDistribution IS
BEGIN
    DECLARE
        CURSOR tickets IS
            SELECT
                s.labelShow
                ,tc.labelTheater
                ,ts.dat
                ,t.labelTicket
                ,COUNT(ts.idTicket) AS distrib
            FROM TICKET_SHOW ts
            JOIN TICKETS t ON ts.idTicket = t.idTicket AND ts.idShow = t.idShow AND ts.idTheaterCompany = t.idTheater
            JOIN SHOWS s ON t.idShow = s.idShow AND t.idTheater = s.idTheater
            JOIN THEATER_COMPANY tc ON s.idTheater = tc.idTheater
            GROUP BY s.labelShow, tc.labelTheater, ts.dat, t.labelTicket
            ORDER BY (s.labelShow) DESC, t.labelTicket DESC;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('La distribution par type de tarif des tickets des différentes représentations est la suivante : ');
        FOR ticket IN tickets LOOP
            DBMS_OUTPUT.PUT_LINE(ticket.labelShow || ', ' || ticket.labelTheater || ', à la date du ' || ticket.dat || ', au tarif de ' || ticket.labelTicket ||
            ' : ' || ticket.distrib || ' places vendue(s) ');
            EXIT WHEN tickets%NOTFOUND;
        END LOOP;
    END;
END;
/

BEGIN
    labelPriceDistribution;
END;
```

```
Statement processed.
La distribution par type de tarif des tickets des différentes représentations est la suivante :
Tonton était bon, Temple Solaire, à la date du 07-07-2013, au tarif de reducedPriceTicket : 4 places vendue(s)
Tonton était bon, Temple Solaire, à la date du 09-09-2021, au tarif de reducedPriceTicket : 2 places vendue(s)
Tonton était bon, Temple Solaire, à la date du 07-07-2013, au tarif de normalPriceTicket : 1 places vendue(s)
Tonton était bon, Temple Solaire, à la date du 09-09-2021, au tarif de normalPriceTicket : 1 places vendue(s)
Les yamnayas, un voyage à travers le temps, Sapien Institute, à la date du 19-03-2015, au tarif de reducedPriceTicket : 5 places vendue(s)
Les yamnayas, un voyage à travers le temps, Sapien Institute, à la date du 23-05-2017, au tarif de reducedPriceTicket : 3 places vendue(s)
Les yamnayas, un voyage à travers le temps, Sapien Institute, à la date du 23-05-2017, au tarif de normalPriceTicket : 3 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 06-06-2013, au tarif de reducedPriceTicket : 3 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 18-03-2016, au tarif de reducedPriceTicket : 6 places vendue(s)
Le braquage du siècle, Café Rock, à la date du 18-03-2016, au tarif de normalPriceTicket : 2 places vendue(s)
Le Roi Soleil, Temple de la Liberté, à la date du 07-06-2017, au tarif de reducedPriceTicket : 6 places vendue(s)
```

- **PROCEDURE showAverageLoadFactor**

Cette procédure sans paramètre sert à afficher le remplissage moyen des salles pour chaque théâtre. Pour cela, on utilise une requête au-dessus.

```
CREATE OR REPLACE PROCEDURE
showAverageLoadFactor IS
BEGIN
    DECLARE
        CURSOR loadFactors IS
            SELECT
                cte.idTheaterHall
                ,hall.labelTheater
                ,ROUND(AVG(load_factor), 2) AS average_load_factor
            FROM (
                SELECT
                    idTheaterHall
                    ,(COUNT(idTicket) / hallCapacity) AS load_factor
                FROM TICKET_SHOW ts
                GROUP BY ts.idShow, ts.idTheaterCompany, ts.dat, ts.hallCapacity, idTheaterHall
            ) cte
            JOIN THEATER_HALL hall ON cte.idTheaterHall = hall.idTheater
            GROUP BY cte.idTheaterHall, hall.labelTheater
            ORDER BY average_load_factor DESC;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Le taux de remplissage pour chaque théâtre est : ');
        FOR lf IN loadFactors LOOP
            DBMS_OUTPUT.PUT_LINE('Théâtre ' || lf.labelTheater || ', remplissage moyen de : 0' || lf.average_load_factor);
            EXIT WHEN loadFactors%NOTFOUND;
        END LOOP;
    END;
END;
/

BEGIN
    showAverageLoadFactor;
END;
```

```
Statement processed.
Le taux de remplissage pour chaque théâtre est :
Théâtre Euismod Enim Etiam, remplissage moyen de : 0.33
Théâtre Temple de la Liberté, remplissage moyen de : 0.24
Théâtre Sapien Institute, remplissage moyen de : 0.16
Théâtre Café Rock, remplissage moyen de : 0.12
Théâtre Temple Solaire, remplissage moyen de : 0.07
```

ACCOUNTING

- **PROCEDURE firstRedBalanceDate**

1. Cette procédure sert à afficher la première date à laquelle le solde d'un théâtre sera dans le rouge (dans l'hypothèse où aucun billet n'est vendu)

```
drop procedure firstRedBalanceDate;

CREATE OR REPLACE PROCEDURE
firstRedBalanceDate(nametheater VARCHAR2) IS
BEGIN
    DECLARE
        CURSOR theaters IS
            SELECT
                MIN(ts.dat) AS firstRedBalanceDate
                ,ts.idTheaterHall
                ,tc.labelTheater
            FROM TICKET_SHOW ts
            JOIN THEATER_HALL hall ON ts.idTheaterHall = hall.idTheater
            JOIN THEATER_COMPANY tc ON hall.idTheater = tc.idTheater
            WHERE tc.labelTheater = nametheater
            GROUP BY ts.idTheaterHall, tc.labelTheater, ts.dat
            HAVING COUNT(ts.idTicket) = (
                SELECT COUNT(idTicket)
                FROM TICKET_SHOW
                GROUP BY idTheaterHall, dat
                HAVING COUNT(idTicket) = 0
            );
    BEGIN
        FOR theater IN theaters LOOP
            DBMS_OUTPUT.PUT_LINE('La première date à laquelle le théâtre' || theater.labelTheater || ' se retrouve dans le rouge,
            avec aucun billet vendu, est : ' || theater.firstRedBalanceDate);
            EXIT WHEN theaters%NOTFOUND;
        END LOOP;
    END;
END;
/


BEGIN
    firstRedBalanceDate('Café Rock');
END;
```

```
Statement processed.
```

- **PROCEDURE firstPermanentlyRedBalanceDate**

2. Cette procédure sert à afficher la première date à laquelle le solde d'un théâtre passera définitivement dans le rouge (dans l'hypothèse où aucun billet n'est vendu, et où les recettes attendues ne compensent pas suffisamment).

```

drop procedure firstPermanentlyRedBalanceDate;

CREATE OR REPLACE PROCEDURE
firstPermanentlyRedBalanceDate(nametheater VARCHAR2, expectedRevenue NUMBER) IS
BEGIN
    DECLARE
        CURSOR theaters IS
            SELECT
                MIN(ts.dat) AS firstPermanentlyRedBalanceDate
                ,ts.idTheaterHall
                ,tc.labelTheater
            FROM TICKET_SHOW ts
            JOIN THEATER_HALL hall ON ts.idTheaterHall = hall.idTheater
            JOIN THEATER_COMPANY tc ON hall.idTheater = tc.idTheater
            WHERE tc.labelTheater = nametheater
            AND tc.budget < expectedRevenue
            GROUP BY ts.idTheaterHall, tc.labelTheater, ts.dat
            HAVING COUNT(ts.idTicket) = (
                SELECT COUNT(idTicket)
                FROM TICKET_SHOW
                GROUP BY idTheaterHall, dat
                HAVING COUNT(idTicket) = 0
            );
    BEGIN
        FOR theater IN theaters LOOP
            DBMS_OUTPUT.PUT_LINE('La première date à laquelle le théâtre' || theater.labelTheater || ' se retrouve dans le rouge
            de façon permanente, avec aucun billet de vendu et avec un revenu en-dessous de ' || expectedRevenue || '
            est : ' || theater.firstPermanentlyRedBalanceDate);
            EXIT WHEN theaters%NOTFOUND;
        END LOOP;
    END;
END;
/


BEGIN
    firstPermanentlyRedBalanceDate('Euismod Enim Etiam', 2000);
END;

```

```
Statement processed.
```

- **PROCEDURE enoughTickets**

3. Cette procédure sert à afficher le nombre de places vendues par un théâtre à différentes dates afin d'éviter les situations précédentes.

```
drop procedure enoughTickets;

CREATE OR REPLACE PROCEDURE
enoughTickets(nametheater VARCHAR2, expectedRevenue NUMBER) IS
BEGIN
    DECLARE
        CURSOR theaters IS
            SELECT
                COUNT(ts.idTicket) AS nbTicketsToSale
                ,tc.idTheater
                ,tc.labelTheater
                ,ts.dat
            FROM TICKET_SHOW ts
            JOIN THEATER_HALL hall ON ts.idTheaterHall = hall.idTheater
            JOIN THEATER_COMPANY tc ON hall.idTheater = tc.idTheater
            WHERE tc.labelTheater = nametheater
            AND tc.budget >= expectedRevenue
            GROUP BY tc.idTheater, tc.labelTheater, ts.dat
            HAVING COUNT(idTicket) > 0;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Afin d''éviter cette situation, le théâtre ' || nametheater || ' vend : ');
        FOR theater IN theaters LOOP
            DBMS_OUTPUT.PUT_LINE(theater.nbTicketsToSale || ' places ');
            EXIT WHEN theaters%NOTFOUND;
        END LOOP;
    END;
END;
/

BEGIN
    enoughTickets('Euismod Enim Etiam', 50000);
END;
```

```
Statement processed.
Afin d'éviter cette situation, le théâtre Euismod Enim Etiam vend :
6 places
9 places
3 places
5 places
4 places
7 places
```

- **PROCEDURE balanceSoldOut1**

4. Cette procédure sert à afficher le solde potentiel de la représentation d''une compagnie de théâtre pour un soir donné, si le spectacle affiche complet.

Ici, nous faisons l'hypothèse que *la compagnie accueille le spectacle qu'elle produit dans sa propre salle.*

Pour les deux autres hypothèses, *'la compagnie exporte le spectacle qu''elle produit dans une autre salle'* et *'le théâtre accueille un spectacle produit par une autre compagnie dans sa salle'*, se reporter aux procédures **balanceSoldOut2** & **balanceSoldOut3** du fichier **requetes.sql**.

```
drop procedure balanceSoldOut1;



CREATE OR REPLACE PROCEDURE
balanceSoldOut1(nameShow VARCHAR2, nametheater VARCHAR2, dateshow DATE) IS
BEGIN
    DECLARE
        CURSOR theaters IS
            SELECT
                s.labelShow, tc.labelTheater, r.dat,
                SUM(r.hallCapacity*25) AS ticketingIncome
                ,(r.comFeesPerRep + r.prodCosts) AS costsIncurred
                ,SUM(r.hallCapacity*25) + (r.comFeesPerRep + r.prodCosts) AS diff
            FROM THEATER_HALL hall
            JOIN REPRESENTATION r ON r.idTheaterHall = hall.idTheater
            JOIN SHOWS s ON s.idShow = r.idShow AND s.idTheater = r.idTheaterCompany
            JOIN THEATER_COMPANY tc ON s.idTheater = tc.idTheater
            WHERE r.idTheaterCompany = r.idTheaterHall
            AND s.labelShow = nameShow AND tc.labelTheater = nametheater AND r.dat = dateshow
            GROUP BY s.labelShow, tc.labelTheater, r.dat, r.comFeesPerRep, r.prodCosts;
    BEGIN
        FOR theater IN theaters LOOP
            DBMS_OUTPUT.PUT_LINE('Le spectacle ' || theater.labelShow || ', de la compagnie productrice ' || theater.labelTheater ||
            ' a, s''il est complet, un solde potentiel de ' || theater.diff || ' au soir du : ' || dateshow);
            EXIT WHEN theaters%NOTFOUND;
            IF theater.diff < 0 THEN
                DBMS_OUTPUT.PUT_LINE('Ce spectacle n''est pas rentable pour la compagnie productrice');
            ELSE
                DBMS_OUTPUT.PUT_LINE('Ce spectacle est rentable pour la compagnie productrice');
            END IF;
        END LOOP;
    END;
END;
/
BEGIN
    balanceSoldOut1('Croc Bagne', 'Euismod Enim Etiam', '23-07-2019');
END;
```

```
Procedure created.


Statement processed.
Le spectacle Croc Bagne, de la compagnie productrice Euismod Enim Etiam a, s'il est complet, un solde potentiel de -1253 au soir du : 23-07-2019
Ce spectacle n'est pas rentable pour la compagnie productrice
```

- **PROCEDURE effectiveBalance1**

5. Cette procédure sert à afficher le solde réel de la représentation d'une compagnie de théâtre pour un soir donné.

Ici, nous faisons l'hypothèse que *la compagnie accueille le spectacle qu'elle produit dans sa propre salle.*

Pour les deux autres hypothèses, *'la compagnie exporte le spectacle qu''elle produit dans une autre salle'* et *'le théâtre accueille un spectacle produit par une autre compagnie dans sa salle'*, se reporter aux procédures **effectiveBalance2** & **effectiveBalance3** du fichier **requetes.sql**.

```
CREATE OR REPLACE PROCEDURE
effectiveBalance1(nameShow VARCHAR2, nametheater VARCHAR2, dateshow DATE) IS
BEGIN
    DECLARE
        CURSOR theaters IS
            SELECT
                s.labelShow, tc.labelTheater, ts.dat,
                SUM(ts.priceTicket) AS ticketingIncome
                ,(r.comFeesPerRep + r.prodCosts) AS costsIncurred
                ,SUM(ts.priceTicket) + (r.comFeesPerRep + r.prodCosts) AS diff
            FROM TICKET_SHOW ts
            JOIN THEATER_HALL hall ON ts.idTheaterHall = hall.idTheater
            JOIN REPRESENTATION r ON r.idTheaterHall = hall.idTheater
            JOIN SHOWS s ON s.idShow = r.idShow AND s.idTheater = r.idTheaterCompany
            JOIN THEATER_COMPANY tc ON s.idTheater = tc.idTheater
            WHERE r.idTheaterCompany = r.idTheaterHall
            AND s.labelShow = nameShow AND tc.labelTheater = nametheater AND ts.dat = dateshow
            GROUP BY s.labelShow, tc.labelTheater, ts.dat, r.comFeesPerRep, r.prodCosts
            ORDER BY costsIncurred ASC
            FETCH FIRST 1 ROWS ONLY;
    BEGIN
        FOR theater IN theaters LOOP
            DBMS_OUTPUT.PUT_LINE('Le spectacle ' || theater.labelShow || ', de la compagnie productrice ' || theater.labelTheater ||
            ' a un solde réel de ' || theater.diff || ' au soir du : ' || dateshow);
            EXIT WHEN theaters%NOTFOUND;
            IF theater.diff < 0 THEN
                DBMS_OUTPUT.PUT_LINE('Ce spectacle n''est pas rentable pour la compagnie productrice');
            ELSE
                DBMS_OUTPUT.PUT_LINE('Ce spectacle est rentable pour la compagnie productrice');
            END IF;
        END LOOP;
    END;
END;
/
BEGIN
    effectiveBalance1('Croc Bagne', 'Euismod Enim Etiam', '23-07-2019');
END;
```

```
Procedure created.


Statement processed.
Le spectacle Croc Bagne, de la compagnie productrice Euismod Enim Etiam a un solde réel de -1633 au soir du : 23-07-2019
Ce spectacle n'est pas rentable pour la compagnie productrice
```

NETWORK

- **PROCEDURE companiesNotAtHome**

Cette procédure sert à afficher les compagnies de théâtre qui ne jouent jamais dans leur propre théâtre.

```
5
6  CREATE OR REPLACE PROCEDURE
7  companiesNotAtHome IS
8  BEGIN
9      DECLARE
10         CURSOR theaters IS
11             SELECT
12                 tc.idTheater
13                 ,tc.labelTheater
14             FROM THEATER_COMPANY tc
15             JOIN SHOWS s ON s.idTheater = tc.idTheater
16             JOIN REPRESENTATION r ON s.idShow = r.idShow AND s.idTheater = r.idTheaterCompany
17             JOIN THEATER_HALL hall ON hall.idTheater = r.idTheaterHall
18             WHERE r.idTheaterHall != r.idTheaterCompany
19             GROUP BY tc.idTheater, tc.labelTheater
20             HAVING tc.idTheater NOT IN (
21                 SELECT
22                     tc.idTheater
23                 FROM THEATER_COMPANY tc
24                 JOIN SHOWS s ON s.idTheater = tc.idTheater
25                 JOIN REPRESENTATION r ON s.idShow = r.idShow AND s.idTheater = r.idTheaterCompany
26                 JOIN THEATER_HALL hall ON hall.idTheater = r.idTheaterHall
27                 WHERE r.idTheaterHall = r.idTheaterCompany
28                 GROUP BY tc.idTheater
29             );
30     BEGIN
31     DBMS_OUTPUT.PUT_LINE('Les compagnies de théâtre qui ne jouent jamais à domicile sont : ');
32         FOR theater IN theaters LOOP
33             DBMS_OUTPUT.PUT_LINE(theater.labelTheater);
34             EXIT WHEN theaters%NOTFOUND;
35         END LOOP;
36     END;
37 END;
38 /
39
40 BEGIN
41     companiesNotAtHome;
42 END;
```

```
Statement processed.
Les compagnies de théâtre qui ne jouent jamais à domicile sont :
Le caroussel du rire
```

- **PROCEDURE companiesFirstShowAtHome**

Cette procédure sert à afficher les compagnies de théâtre qui jouent systématiquement leur première représentation dans leur propre salle.

```
1
2  drop procedure companiesFirstShowAtHome;
3
4
5  CREATE OR REPLACE PROCEDURE
6  companiesFirstShowAtHome IS
7  BEGIN
8      DECLARE
9          CURSOR theaters IS
10             select
11             LABELTHEATER,
12             dat
13             from
14             representation r,
15             THEATER_COMPANY t
16             where
17             t.IDTHEATER = r.IDTHEATERCOMPANY
18             and r.IDTHEATERCOMPANY = r.IDTHEATERHALL
19             and r.dat = (
20                 select
21                 min (dat)
22                 from
23                 representation
24                 where
25                 IDTHEATERCOMPANY = r.IDTHEATERCOMPANY
26             );
27
28     BEGIN
29     DBMS_OUTPUT.PUT_LINE('Les compagnies de théâtre qui jouent systématiquement leur première représentation à domicile sont : ');
30         FOR theater IN theaters LOOP
31             DBMS_OUTPUT.PUT_LINE(theater.labelTheater);
32             EXIT WHEN theaters%NOTFOUND;
33         END LOOP;
34     END;
35 END;
36 /
37
38
39 BEGIN
40     companiesFirstShowAtHome;
41 END;
42
```

```
Statement processed.
Les compagnies de théâtre qui jouent systématiquement leur première représentation à domicile sont :
Temple de la Liberté
Euismod Enim Etiam
```

- **PROCEDURE companiesFirstShowOutside**

Cette procédure sert à afficher les compagnies de théâtre qui jouent systématiquement leur première représentation dans une salle extérieure.

```
drop procedure companiesFirstShowOutside;


CREATE OR REPLACE PROCEDURE
companiesFirstShowOutside IS
BEGIN
    DECLARE
        CURSOR theaters IS
            select
            t.LABELTHEATER,
            dat
            from
            representation r,
            THEATER_COMPANY t
            where
            t.IDTHEATER = r.IDTHEATERCOMPANY
            and r.IDTHEATERCOMPANY <> r.IDTHEATERHALL
            and r.dat = (
                select
                min (dat)
                from
                representation
                where
                IDTHEATERCOMPANY = r.IDTHEATERCOMPANY
            );

    BEGIN
    DBMS_OUTPUT.PUT_LINE('Les compagnies de théâtre qui jouent systématiquement leur première représentation à l''extérieur sont : ');
        FOR theater IN theaters LOOP
            DBMS_OUTPUT.PUT_LINE(theater.labelTheater);
            EXIT WHEN theaters%NOTFOUND;
        END LOOP;
    END;
END;
/

BEGIN
    companiesFirstShowOutside;
END;

```

```
Statement processed.
Les compagnies de théâtre qui jouent systématiquement leur première représentation à l'extérieur sont :
Café Rock
Sapien Institute
Temple Solaire
```

- **PROCEDURE nbOfRep**

Cette procédure sert à afficher le nombre de représentations d'un même spectacle pendant une période de temps donnée.

```
drop procedure nbOfRep;

CREATE OR REPLACE PROCEDURE
nbOfRep(dateshow1 DATE, dateshow2 DATE) IS
BEGIN
    DECLARE
        CURSOR seats IS
            SELECT
                s.labelShow
                ,COUNT(r.idShow) AS nbOfRep
            FROM REPRESENTATION r
            JOIN SHOWS s ON r.idShow = s.idShow AND r.idTheaterCompany = s.idTheater
            WHERE r.dat BETWEEN dateshow1 AND dateshow2
            GROUP BY s.labelShow
            ORDER BY COUNT(r.idShow) DESC
            FETCH FIRST 3 ROWS ONLY;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Les spectacles les plus joués entre le ' || dateshow1 || ' et le ' || dateshow2 || ' sont : ');
        FOR seat IN seats LOOP
            DBMS_OUTPUT.PUT_LINE(seat.labelShow || ', avec ' || seat.nbOfRep || ' représentations');
            EXIT when seats%NOTFOUND;
        END LOOP;
    END;
END;
/


BEGIN
    nbOfRep('29-02-2012', '17-04-2018');
END;
```

```
Statement processed.
Les spectacles les plus joués entre le 29-02-2012 et le 17-04-2018 sont :
Les yamnayas, un voyage à travers le temps, avec 2 représentations
Le Roi Soleil, avec 2 représentations
Le braquage du siècle, avec 2 représentations
```

- **PROCEDURE nbOfSeatstoSell**

Cette procédure sert à afficher le nombre de spectateurs potentiels d'un même spectacle pendant une période de temps donnée.

```
drop procedure nbOfSeatstoSell;

CREATE OR REPLACE PROCEDURE
nbOfSeatstoSell(dateshow1 DATE, dateshow2 DATE) IS
BEGIN
    DECLARE
        CURSOR seats IS
            SELECT
                s.labelShow
                ,SUM(r.hallCapacity) AS nbOfSeats
            FROM REPRESENTATION r
            JOIN SHOWS s ON r.idShow = s.idShow AND r.idTheaterCompany = s.idTheater
            WHERE r.dat BETWEEN dateshow1 AND dateshow2
            GROUP BY s.labelShow
            ORDER BY SUM(r.hallCapacity) DESC
            FETCH FIRST 5 ROWS ONLY;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Les spectacles potentiellement les plus populaires entre le ' || dateshow1 || ' et le ' || dateshow2 || ' sont : ');
        FOR seat IN seats LOOP
            DBMS_OUTPUT.PUT_LINE(seat.labelShow || ', avec ' || seat.nbOfSeats || ' places à vendre');
            EXIT when seats%NOTFOUND;
        END LOOP;
    END;
END;
/


BEGIN
    nbOfSeatstoSell('29-02-2012', '17-04-2018');
END;
```

```
Statement processed.
Les spectacles potentiellement les plus populaires entre le 29-02-2012 et le 17-04-2018 sont :
Le braquage du siècle, avec 108 places à vendre
Auguste c'est le minimum, avec 54 places à vendre
Les yamnayas, un voyage à travers le temps, avec 42 places à vendre
L'entrée des artistes, avec 33 places à vendre
Fer et Acier, avec 33 places à vendre
```

- **PROCEDURE nbOfSeatsSold**

Cette procédure sert à afficher le nombre de sièges vendus pour un même spectacle pendant une période de temps donnée.

```
CREATE OR REPLACE PROCEDURE
nbOfSeatsSold(dateshow1 DATE, dateshow2 DATE) IS
BEGIN
    DECLARE
        CURSOR seats IS
            SELECT
                s.idShow,
                s.labelShow
                ,tc.labelTheater
                ,COUNT(ts.idTicket) AS nbMaxOfSeatsSold
            FROM REPRESENTATION r
            JOIN SHOWS s ON r.idShow = s.idShow AND r.idTheaterCompany = s.idTheater
            JOIN TICKET_SHOW ts ON s.idShow = ts.idShow AND s.idTheater = ts.idTheaterCompany
            JOIN THEATER_COMPANY tc ON s.idTheater = tc.idTheater
            WHERE r.dat BETWEEN '29-02-2012' AND '17-04-2018'
            GROUP BY s.idShow, s.labelShow, tc.labelTheater
            ORDER BY COUNT(ts.idTicket) DESC
            FETCH FIRST 5 ROWS ONLY;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Les spectacles les plus populaires entre le ' || dateshow1 || ' et le ' || dateshow2 || ' sont : ');
        FOR seat IN seats LOOP
            DBMS_OUTPUT.PUT_LINE(seat.labelShow || ', (' || seat.labelTheater || '), avec ' || seat.nbMaxOfSeatsSold || ' places vendues');
            EXIT when seats%NOTFOUND;
        END LOOP;
    END;
END;
/

BEGIN
    nbOfSeatsSold('29-02-2012', '17-04-2018');
END;
```

```
Statement processed.
Les spectacles les plus populaires entre le 29-02-2012 et le 17-04-2018 sont :
Le braquage du siècle, (Café Rock), avec 22 places vendues
Les yamnayas, un voyage à travers le temps, (Sapien Institute), avec 22 places vendues
Le Roi Soleil, (Temple de la Liberté), avec 18 places vendues
La roue solaire, (Temple de la Liberté), avec 12 places vendues
Tonton était bon, (Temple Solaire), avec 8 places vendues
```