



Project SQL fin de module

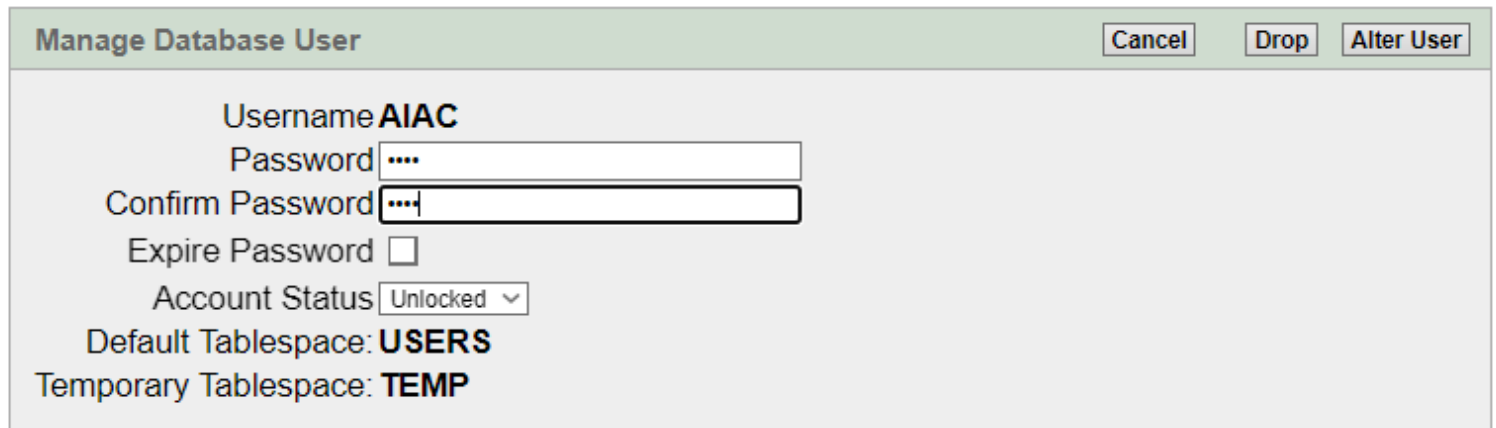
Realiser par :

Abdellah Lamine & EL -Azzaouy

Encadrée par :

Mohamed El wafiq

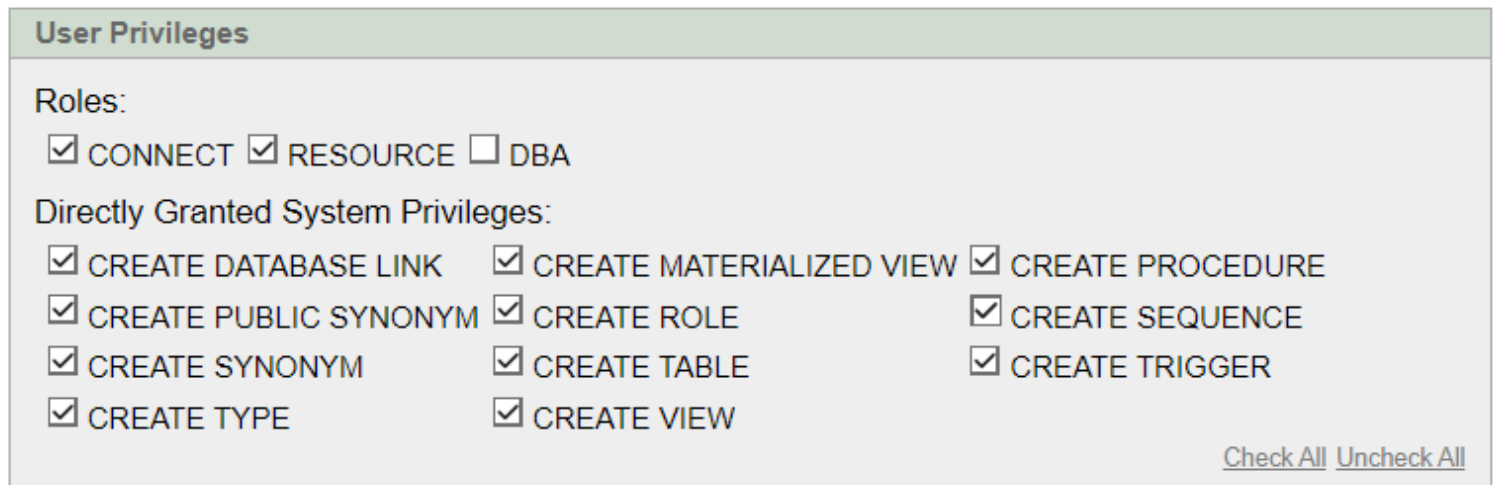
pour commencer on doit d'abord crée un nouveau utilisateur
est l'appeler AIAC avec tous les privilèges



The 'Manage Database User' dialog box is shown with the following fields and values:

- Username: **AIAC**
- Password: [masked with dots]
- Confirm Password: [masked with dots]
- Expire Password: ☐
- Account Status: Unlocked (dropdown)
- Default Tablespace: **USERS**
- Temporary Tablespace: **TEMP**

Buttons at the top right: Cancel, Drop, Alter User.



The 'User Privileges' dialog box shows the following settings:

Roles:

- ☒ CONNECT
- ☒ RESOURCE
- ☐ DBA

Directly Granted System Privileges:

| | | |
|---|--|--|
| <input checked="" type="checkbox"/> CREATE DATABASE LINK | <input checked="" type="checkbox"/> CREATE MATERIALIZED VIEW | <input checked="" type="checkbox"/> CREATE PROCEDURE |
| <input checked="" type="checkbox"/> CREATE PUBLIC SYNONYM | <input checked="" type="checkbox"/> CREATE ROLE | <input checked="" type="checkbox"/> CREATE SEQUENCE |
| <input checked="" type="checkbox"/> CREATE SYNONYM | <input checked="" type="checkbox"/> CREATE TABLE | <input checked="" type="checkbox"/> CREATE TRIGGER |
| <input checked="" type="checkbox"/> CREATE TYPE | <input checked="" type="checkbox"/> CREATE VIEW | |

Buttons at the bottom right: [Check All](#) [Uncheck All](#)

après il suffit de se connecter est commencer les exécutions

```
SQL> connect AIAC
Enter password:
Connected.
SQL> _
```

1. Créez les tables à partir des informations contenues dans les tableaux ci-dessous. Choisissez les types de données appropriés et veillez à ajouter des contraintes d'intégrité.

a. Nom de la table : MEMBER

| Nom de colonne | MEMBER_ID | LAST_NAME | FIRST_NAME | ADDRESS | CITY | PHONE | JOIN_DATE |
|-------------------|-----------|-----------|------------|----------|----------|----------|--------------|
| Type de clé | PK | | | | | | |
| Null/Unique | NN,U | NN | | | | | NN |
| Valeur par défaut | | | | | | | Date système |
| Type de données | NUMBER | VARCHAR2 | VARCHAR2 | VARCHAR2 | VARCHAR2 | VARCHAR2 | DATE |
| Longueur | 10 | 25 | 25 | 100 | 30 | 15 | |

Pour faire cela on va lancer la requête suivant :

```
CREATE TABLE member (  
  NUMBER_id number PRIMARY KEY NOT NULL,  
  last_name VARCHAR2(50) NOT NULL,  
  first_name VARCHAR2(50),  
  address VARCHAR2(100),  
  phone VARCHAR2(20),  
  join_date DATE DEFAULT SYSDATE NOT NULL);
```

```
SQL> CREATE TABLE member (  
2   NUMBER_id number PRIMARY KEY NOT NULL,  
3   last_name VARCHAR2(50) NOT NULL,  
4   first_name VARCHAR2(50),  
5   address VARCHAR2(100),  
6   phone VARCHAR2(20),  
7   join_date DATE DEFAULT SYSDATE NOT NULL  
8 );  
  
Table created.
```

On remarque que notre table est créée avec succès !

Pour la 2eme tableau :

b. Nom de la table : TITLE

| | | | | | | |
|-----------------|----------|----------|-------------|--------------------|--|--------------|
| Nom de colonne | TITLE_ID | TITLE | DESCRIPTION | RATING | CATEGORY | RELEASE_DATE |
| Type de clé | PK | | | | | |
| Null/Unique | NN,U | NN | NN | | | |
| Contrôle | | | | G, PG, R, NC17, NR | DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY | |
| Type de données | NUMBER | VARCHAR2 | VARCHAR2 | VARCHAR2 | VARCHAR2 | DATE |
| Longueur | 10 | 60 | 400 | 4 | 20 | |

On le crée avec la requête :

```
CREATE TABLE title (  
  title_id INT NOT NULL PRIMARY KEY,  
  title VARCHAR2(60) NOT NULL,  
  description VARCHAR2(400) NOT NULL,  
  rating VARCHAR2(4),  
  category VARCHAR2(20),  
  release_date DATE,  
  CONSTRAINT controle_category CHECK (category IN ('DRAMA', 'COMEDY', 'ACTION', 'CHILD', 'SCIFI', 'DOCUMENTARY')),  
  CONSTRAINT controle_rating CHECK (rating IN ('G', 'PG', 'R', 'NC17', 'NR'))  
);
```

```
SQL> CREATE TABLE title (  
  2  title_id INT NOT NULL PRIMARY KEY,  
  3  title VARCHAR2(60) NOT NULL,  
  4  description VARCHAR2(400) NOT NULL,  
  5  rating VARCHAR2(4),  
  6  category VARCHAR2(20),  
  7  release_date DATE,  
  8  CONSTRAINT controle_category CHECK (category IN ('DRAMA', 'COMEDY', 'ACTION', 'CHILD', 'SCIFI', 'DOCUMENTARY')),  
  9  CONSTRAINT controle_rating CHECK (rating IN ('G', 'PG', 'R', 'NC17', 'NR'))  
 10 );
```

Table created.

Le tableau a été créé avec succès !

La 3eme tableau :

c. Nom de la table : TITLE_COPY

| | | | |
|---|---------|----------|---|
| Nom de colonne | COPY_ID | TITLE_ID | STATUS |
| Type de clé | PK | PK,FK | |
| Null/Unique | NN,U | NN,U | NN |
| Contrôle | | | AVAILABLE, DESTROYED, RENTED, RESERVED |
| Table de référence de la clé étrangère | | TITLE | |
| Colonne de référence de la clé étrangère | | TITLE_ID | |
| Type de données | NUMBER | NUMBER | VARCHAR2 |
| Longueur | 10 | 10 | 15 |

On lance la requête suivante pour la création de ce table.

```
CREATE TABLE TITLE_COPY (  
  COPY_ID NUMBER PRIMARY KEY NOT NULL,  
  TITLE_ID NUMBER REFERENCES TITLE(TITLE_ID),  
  STATUS VARCHAR2(15) NOT NULL,  
  CONSTRAINT Controle CHECK (STATUS IN ('AVAILABLE', 'DESTROYED',  
  'RENTED', 'RESERVED'))  
);
```

```
ALTER TABLE title_copy
```

```
ADD CONSTRAINT unique_copy UNIQUE (copy_id, title_id);
```

```
SQL> CREATE TABLE TITLE_COPY (  
  2   COPY_ID NUMBER PRIMARY KEY NOT NULL,  
  3   TITLE_ID NUMBER REFERENCES TITLE(TITLE_ID),  
  4   STATUS VARCHAR2(15) NOT NULL,  
  5   CONSTRAINT Controle CHECK (STATUS IN ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED'))  
  6 );
```

```
Table created.
```

Le tableau a été créé avec succès !

Pour le tableau 4 : rental

d. Nom de la table : RENTAL

| Nom de colonne | BOOK_ DATE | MEMBER_ ID | COPY_ ID | ACT_RET_ DATE | EXP_RET_ DATE | TITLE_ ID |
|--|---------------|---------------|-------------|------------------|------------------------|--------------|
| Type de clé | PK | PK,FK1 | PK,FK2 | | | PK,FK2 |
| Valeur par défaut | Date système | | | | Date système + 2 jours | |
| Table de référence de la clé étrangère | | MEMBER | TITLE_COPY | | | TITLE_COPY |
| Colonne de référence de la clé étrangère | | MEMBER_ID | COPY_ID | | | TITLE_ID |
| Type de données | DATE | NUMBER | NUMBER | DATE | DATE | NUMBER |
| Longueur | | 10 | 10 | | | 10 |

On lance la requête suivante pour la création de ce table.

CREATE TABLE rental (

Book_DATE DATE DEFAULT SYSDATE,

NUMBER_id NUMBER,

copy_id NUMBER,

ACT_RET_DATE DATE DEFAULT SYSDATE + INTERVAL '2' DAY,

EXP_RET_DATE DATE DEFAULT SYSDATE + INTERVAL '2' DAY,

title_id NUMBER,

PRIMARY KEY (Book_DATE, NUMBER_id, copy_id, title_id),

FOREIGN KEY (NUMBER_id) REFERENCES member (NUMBER_id),

FOREIGN KEY (copy_id) REFERENCES TITLE_COPY (COPY_ID),

FOREIGN KEY (title_id) REFERENCES TITLE (title_id)

);

```
SQL> CREATE TABLE rental (  
2   Rook_DATE DATE DEFAULT SYSDATE,  
3   NUMBER_id NUMBER,  
4   copy_id NUMBER,  
5   ACT_RET_DATE DATE DEFAULT SYSDATE + INTERVAL '2' DAY,  
6   EXP_RET_DATE DATE DEFAULT SYSDATE + INTERVAL '2' DAY,  
7   title_id NUMBER,  
8   PRIMARY KEY (Rook_DATE, NUMBER_id, copy_id, title_id),  
9   FOREIGN KEY (NUMBER_id) REFERENCES member ( NUMBER_id),  
10  FOREIGN KEY (copy_id) REFERENCES TITLE_COPY (COPY_ID),  
11  FOREIGN KEY (title_id) REFERENCES TITLE (title_id)  
12 );  
  
Table created.
```

Le tableau a été créé avec succès !

Finalement pour la dernière table RESERVATION :

e. Nom de la table : RESERVATION

| Nom de colonne | RES_ DATE | MEMBER_ ID | TITLE_ ID |
|--|--------------|---------------|--------------|
| Type de clé | PK | PK,FK1 | PK,FK2 |
| Null/Unique | NN,U | NN,U | NN |
| Table de référence de la clé étrangère | | MEMBER | TITLE |
| Colonne de référence de la clé étrangère | | MEMBER_ID | TITLE_ID |
| Type de données | DATE | NUMBER | NUMBER |
| Longueur | | 10 | 10 |

On le crée par :

CREATE TABLE reservation (

res_date DATE NOT NULL UNIQUE,

NUMBER_id INT NOT NULL,

title_id INT NOT NULL,

PRIMARY KEY (res_date, NUMBER_id, title_id),

```
FOREIGN KEY (NUMBER_id) REFERENCES member(NUMBER_id),
```

```
FOREIGN KEY (title_id) REFERENCES TITLE(title_id));
```

```
SQL> CREATE TABLE reservation (  
2   res_date DATE NOT NULL UNIQUE,  
3   NUMBER_id INT NOT NULL,  
4   title_id INT NOT NULL,  
5   PRIMARY KEY (res_date, NUMBER_id, title_id),  
6   FOREIGN KEY (NUMBER_id) REFERENCES member(NUMBER_id),  
7   FOREIGN KEY (title_id) REFERENCES TITLE(title_id)  
8 );
```

```
Table created.
```

On a alors crée tous les tableau nécessaire il reste qu'a vérifier l'existence de ces tableaux.

Pour ce faire on utilise cette requête :

```
SELECT object_name
```

```
FROM user_objects
```

```
WHERE object_type = 'TABLE'
```

```
AND created >= SYSDATE - 7
```

```
ORDER BY created DESC;
```

cette requête sélectionne les noms des tables de l'utilisateur courant qui ont été créées au cours des 7 derniers jours, triées par date de création décroissante.

```
SQL> SELECT object_name  
2   FROM user_objects  
3   WHERE object_type = 'TABLE'  
4     AND created >= SYSDATE - 7  
5   ORDER BY created DESC;
```

```
OBJECT_NAME
```

```
-----
```

```
RESERVATION
```

```
RENTAL
```

```
TITLE_COPY
```

```
TITLE
```

```
MEMBER
```

On remarque que nous tableaux sont bien crée

Pour afficher les tableaux créés avec leurs contraintes on lance la requête qui sélectionne les noms des contraintes, les noms des tables et les noms des colonnes associées aux contraintes pour les tables créées au cours des 7 derniers jours, triées par date de création décroissante.

```
SELECT UC.CONSTRAINT_NAME, UC.TABLE_NAME, UCC.COLUMN_NAME
FROM USER_CONSTRAINTS UC
JOIN USER_OBJECTS UO ON UC.TABLE_NAME = UO.OBJECT_NAME
JOIN USER_CONS_COLUMNS UCC ON UC.CONSTRAINT_NAME =
UCC.CONSTRAINT_NAME
WHERE UO.OBJECT_TYPE = 'TABLE'
AND UO.CREATED >= SYSDATE - 7
ORDER BY UO.CREATED DESC;
```

| CONSTRAINT_NAME | TABLE_NAME | COLUMN_NAME |
|-------------------|-------------|-------------|
| SYS_C004018 | RESERVATION | NUMBER_ID |
| SYS_C004017 | RESERVATION | RES_DATE |
| SYS_C004016 | RESERVATION | TITLE_ID |
| SYS_C004016 | RESERVATION | NUMBER_ID |
| SYS_C004019 | RESERVATION | TITLE_ID |
| SYS_C004015 | RESERVATION | TITLE_ID |
| SYS_C004014 | RESERVATION | NUMBER_ID |
| SYS_C004013 | RESERVATION | RES_DATE |
| SYS_C004016 | RESERVATION | RES_DATE |
| SYS_C004009 | RENTAL | BOOK_DATE |
| SYS_C004012 | RENTAL | TITLE_ID |
| SYS_C004011 | RENTAL | COPY_ID |
| SYS_C004010 | RENTAL | NUMBER_ID |
| SYS_C004009 | RENTAL | TITLE_ID |
| SYS_C004009 | RENTAL | COPY_ID |
| SYS_C004009 | RENTAL | NUMBER_ID |
| SYS_C004008 | TITLE_COPY | TITLE_ID |
| CONTROLE | TITLE_COPY | STATUS |
| SYS_C004005 | TITLE_COPY | STATUS |
| SYS_C004004 | TITLE_COPY | COPY_ID |
| SYS_C004007 | TITLE_COPY | COPY_ID |
| CONTROLE_RATING | TITLE | RATING |
| SYS_C004000 | TITLE | DESCRIPTION |
| CONTROLE_CATEGORY | TITLE | CATEGORY |

3. Créez des séquences pour identifier de façon unique chaque ligne des tables MEMBER et TITLE.

a- Pour la table MEMBER, le premier numéro de membre doit être 101.

N'autorisez pas la mise en mémoire cache des valeurs et nommez la séquence MEMBER_ID_SEQ.

```
SQL> CREATE SEQUENCE MEMBER_ID_SEQ
2      START WITH 101
3      INCREMENT BY 1
4      CACHE 2;
```

b-Pour la table TITLE, le premier numéro de titre doit être 92. N'autorisez pas la mise en mémoire cache des valeurs et nommez la séquence TITLE_ID_SEQ.

```
SQL> CREATE SEQUENCE TITLE_ID_SEQ
2      START WITH 92
3      INCREMENT BY 1
4      CACHE 2;
```

c-Vérifiez que les séquences existent dans le dictionnaire de données.

```
SQL> SELECT sequence_name, increment_by, last_number
2 FROM user_sequences
3 WHERE sequence_name IN ('MEMBER_ID_SEQ', 'TITLE_ID_SEQ');
```


| SEQUENCE_NAME | INCREMENT_BY | LAST_NUMBER |
|---------------|--------------|-------------|
| MEMBER_ID_SEQ | 1 | 101 |
| TITLE_ID_SEQ | 1 | 92 |

4. Ajoutez des données aux tables. Créez un script pour chaque ensemble de données à ajouter.

a. Ajoutez des titres de films dans la table TITLE. Ecrivez un script pour saisir les informations relatives aux films et enregistrez les instructions dans un script nommé lab14_4a.sql. Utilisez les séquences pour identifier chaque titre de façon unique. Entrez les dates de sortie au format DD-MON-YYYY. N'oubliez pas que les apostrophes présentes dans un champ alphanumérique sont soumises à un traitement spécial.

- Pour créer un script on devra utiliser soit **ED** écrire nous script l'enregistrer puis l'appeler ou bien appeler n'importe quel script .sql avec son path absolu dans le système.
- On tape ed un block note page s'affiche pour écrire nous script

```
SQL> ed lab14_4a
```

 Sans titre - Bloc-notes

Fichier Edition Format Affichage Aide

On écrit donc le script qui nous permet de remplir les colonnes de tableau TITLE avec les inputs de l'utilisateur.

Fichier Edition Format Affichage Aide

```
INSERT INTO TITLE (title_id, title, description, rating, category, release_date)
VALUES (TITLE_ID_SEQ.NEXTVAL, '&title', '&description', '&rating', '&category', TO_DATE('&date', 'DD-MONTH-YYYY'));
```

Après enregistrement de ce script on l'appelle par @script_name dans notre CLI

Les instructions de ce script seront exécutées.

```
SQL> @"C:\Users\hp\Desktop\lab14_4a.sql"
Enter value for title: Willie and Christmas Too
Enter value for description: All Willie's list for Sanata, but willie has yet to add his own wish list.
Enter value for rating: G
Enter value for category: CHILD
Enter value for date: 05-october-1995
old 2: VALUES (TITLE_ID_SEQ.NEXTVAL, '&title', '&description', '&rating', '&category', TO_DATE('&date', 'DD-MONTH-YYYY'))
new 2: VALUES (TITLE_ID_SEQ.NEXTVAL, 'Willie and Christmas Too', 'All Willie's list for Sanata, but willie has yet to add his own wish list.', 'G', 'CHILD', TO_DATE('05-october-1995', 'DD-MONTH-YYYY'))
1 row created.
```

Dans « willie's » on doit ajouter un double « ' » pour échapper les erreurs

Il reste que vérifier si la line existe

```
select * from title;
```

Results Explain Describe Saved SQL History

| TITLE_ID | TITLE | DESCRIPTION | RATING | CATEGORY | RELEASE_DATE |
|----------|-------------------------|--|--------|----------|--------------|
| 92 | Willie and Chrismas Too | All Willie's list for Sanata, but willie has yet to add his own wish list. | G | CHILD | 05-OCT-95 |

Après avoir effectuer la même chose pour le reste des donnes . ca vous dire appeler le script et saisir les donner on se termine par le tableau title de cetter form.

```
select * from title;
```

Results Explain Describe Saved SQL History

| TITLE_ID | TITLE | DESCRIPTION | RATING | CATEGORY | RELEASE_DATE |
|----------|-------------------------|---|--------|----------|--------------|
| 92 | Willie and Chrismas Too | All Willie's list for Sanata, but willie has yet to add his own wish list. | G | CHILD | 05-OCT-95 |
| 93 | Alien again | yet another installation of science fiction history . can the heroine save the planet from the alien life form? | R | SCIFI | 19-MAY-95 |
| 94 | THE Glob | A meteor crashes near a small American town and unleashes carvorous goo in this classic. | NR | SCIFI | 12-AUG-95 |
| 96 | My Day Off | with a little luck and a lot of ingenuity , a teenager skips school for a day in new york. | PG | COMEDY | 12-JUL-95 |
| 97 | Miracles on ice | A six-year-old has doubts about santa claus, but she discovers that miracles really do existe. | PG | DRAMA | 12-SEP-95 |
| 99 | Soda Gang | After discovering a cache of drugs , a young couple find themselves pitted against a vicious gang. | NR | ACTION | 01-JUN-95 |

b. Ajoutez des données à la table MEMBER. Placez les instructions d'insertion dans un script nommé lab14_4b.sql, puis exécutez-les. Veillez à utiliser la séquence pour ajouter les numéros de membre.

On fusant la même chose que avec la question précédent mais cette fois sure le table membre en créant le script la14_4b.sql

```
INSERT INTO member (NUMBER_id,last_name ,first_name, address
,phone,join_date,city) VALUES (MEMBER_ID_SEQ.NEXTVAL, '&last_name',
'&first_name', '&adress', '&phone', TO_DATE('&join_date', 'DD-MONTH-
YYYY'),'&city');
```

On appellent le script pour crée les lignes de tableau member on se termine a la fin par le résultat suivants :

```
select * from member;
```

Results Explain Describe Saved SQL History

| NUMBER_ID | LAST_NAME | FIRST_NAME | ADDRESS | PHONE | JOIN_DATE | CITY |
|-----------|--------------|------------|-----------------|--------------|-----------|------------|
| 103 | Velasquez | carmen | 283 king street | 206-899-6666 | 08-MAR-90 | seattle |
| 104 | ngao | laDoris | 5 Modrany | 254-852-5764 | 17-JUN-91 | Bratislava |
| 106 | Nagayama | Midori | 68 Via Centrale | 254-852-5764 | 17-JUN-91 | sao paulo |
| 107 | Quick-to-See | Mark | 6921 King Way | 63-559-7777 | 07-APR-90 | Lagos |
| 108 | Ropeburn | Audry | 86 Chu Street | 41-559-87 | 18-JAN-91 | HONG KONG |
| 109 | Urguhart | Molly | 3035 Laurier | 418-542-9988 | 18-JAN-91 | Quebec |

6 rows returned in 0.00 seconds [CSV Export](#)

c. Ajoutez les copies de films suivantes dans la table TITLE_COPY :

| Title | Copy_Id | Status |
|--------------------------|---------|-----------|
| Willie and Christmas Too | 1 | AVAILABLE |
| Alien Again | 1 | AVAILABLE |
| | 2 | RENTED |
| The Glob | 1 | AVAILABLE |
| My Day Off | 1 | AVAILABLE |
| | 2 | AVAILABLE |
| | 3 | RENTED |
| Miracles on Ice | 1 | AVAILABLE |
| Soda Gang | 1 | AVAILABLE |

On ajoutra le title par la requete :

```
insert into title_copy(copy_id,title_id,status)
```

```
values('&copy_id','&title_id','&status');
```

on saisira donc les valeurs voulez

```
SQL> insert into title_copy(copy_id,title_id,status)values('&copy_id','&title_id','&status');
Enter value for copy_id: 1
Enter value for title_id: 94
Enter value for status: AVAILABLE
old 1: insert into title_copy(copy_id,title_id,status)values('&copy_id','&title_id','&status')
new 1: insert into title_copy(copy_id,title_id,status)values('1','94','AVAILABLE')
```

le table title_id est donc plain

| COPY_ID | TITLE_ID | STATUS |
|---------|----------|-----------|
| 1 | 92 | AVAILABLE |
| 1 | 93 | AVAILABLE |
| 1 | 94 | AVAILABLE |
| 1 | 96 | AVAILABLE |
| 1 | 97 | AVAILABLE |
| 1 | 99 | AVAILABLE |

d. Ajoutez les locations suivantes dans la table RENTAL :

| Title_ Id | Copy_ Id | Member_ Id | Book date | Exp Ret Date | Act Ret Date |
|-----------|----------|------------|----------------|--------------|--------------|
| 92 | 1 | 101 | Il y a 3 jours | Hier | Avant-hier |
| 93 | 2 | 101 | Hier | Demain | |
| 95 | 3 | 102 | Avant-hier | Aujourd'hui | |
| 97 | 1 | 106 | Il y a 4 jours | Avant-hier | Avant-hier |

On ajouteras ces enregistrements par la requête :

```
insert into
rental(BOOK_DATE,number_id,copy_id,act_ret_date,exp_ret_date,title_id)
values(&BOOK_DATE,'&number_id','&copy_id',&act_ret_date,&exp_ret_date,'
&title_id');
```

il faut faire attention maintenant que lors de la saisir des date il faut entrer

(SYSDATE-<Numbers des jours a substituer pour mettre la date correct >) et
aussi saisir uniquement de valeurs qui son foreign keys existants déjà pour ne
pas avoir des erreurs ,

Donnant la premier exemple

```
SQL> insert into rental(BOOK_DATE,number_id,copy_id,act_ret_date,exp_ret_date,title_id) values(&BOOK_DATE,
'&number_id','&copy_id',&act_ret_date,&exp_ret_date,'&title_id');
Enter value for book_date: (SYSDATE-3)
Enter value for number_id: 103
Enter value for copy_id: 1
Enter value for act_ret_date: (SYSDATE-2)
Enter value for exp_ret_date: (SYSDATE-1)
Enter value for title_id: 92
old 1: insert into rental(BOOK_DATE,number_id,copy_id,act_ret_date,exp_ret_date,title_id) values(&BOOK_D
ATE,'&number_id','&copy_id',&act_ret_date,&exp_ret_date,'&title_id')
new 1: insert into rental(BOOK_DATE,number_id,copy_id,act_ret_date,exp_ret_date,title_id) values((SYSDAT
E-3),'103','1',(SYSDATE-2),(SYSDATE-1),'92')
```

On faisant commit est on lance la selectionement sure la table rental on trouve bien notre enregistrement ;

| select * from rental; | | | | | |
|--|-----------|---------|--------------|--------------|----------|
| Results Explain Describe Saved SQL History | | | | | |
| BOOK_DATE | NUMBER_ID | COPY_ID | ACT_RET_DATE | EXP_RET_DATE | TITLE_ID |
| 13-JUL-23 | 103 | 1 | 14-JUL-23 | 15-JUL-23 | 92 |

On faisant donc la même chose pour les autres enregistrement

Est voila le resultat final de tableau rental :

| select * from rental; | | | | | |
|--|-----------|---------|--------------|--------------|----------|
| Results Explain Describe Saved SQL History | | | | | |
| BOOK_DATE | NUMBER_ID | COPY_ID | ACT_RET_DATE | EXP_RET_DATE | TITLE_ID |
| 13-JUL-23 | 103 | 1 | 14-JUL-23 | 15-JUL-23 | 92 |
| 15-JUL-23 | 103 | 2 | - | 17-JUL-23 | 93 |
| 14-JUL-23 | 104 | 3 | - | 16-JUL-23 | 94 |
| 12-JUL-23 | 106 | 1 | 14-JUL-23 | 14-JUL-23 | 97 |

4 rows returned in 0.00 seconds [CSV Export](#)

6. Modifiez les données des tables.

a- Ajoutez un nouveau titre, "Interstellar Wars", film de science-fiction classé PG. Sa date de sortie est le 07-JUL-77 et sa description est la suivante : "Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?". Dupliquez l'enregistrement pour créer deux copies.

+ d'abord on ajout l'enregistrement

Autrefois on utilise le script lab14_4a.sql

Voici les requêtes lancer :

```
SQL> @"C:\Users\hp\Desktop\lab14_4a.sql"
Enter value for title: Interstellar Wars
Enter value for description: Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?

Enter value for rating: PG
Enter value for category: SCIFI
Enter value for date: 07-JUL-77
old   2: VALUES (TITLE_ID_SEQ.NEXTVAL, '&title', '&description', '&rating', '&category', TO_DATE('&date', 'DD-MONTH-YYY
Y'))
new   2: VALUES (TITLE_ID_SEQ.NEXTVAL, 'Interstellar Wars', 'Futuristic interstellar action movie. Can the rebels save
the humans from the evil empire?', 'PG', 'SCIFI', TO_DATE('07-JUL-77', 'DD-MONTH-YYYY'))

1 row created.

SQL> commit;

Commit complete.

SQL> desc title_copy
Name                                     Null?    Type
-----
COPY_ID                                NOT NULL NUMBER
TITLE_ID                                NUMBER
STATUS                                 NOT NULL VARCHAR2(15)

SQL> @"C:\Users\hp\Desktop\lab14_4a.sql"
Enter value for title: Interstellar Wars
Enter value for description: Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?

Enter value for rating: PG
Enter value for category: SCIFI
Enter value for date: 07-JUL-77
old   2: VALUES (TITLE_ID_SEQ.NEXTVAL, '&title', '&description', '&rating', '&category', TO_DATE('&date', 'DD-MONTH-YYY
Y'))
new   2: VALUES (TITLE_ID_SEQ.NEXTVAL, 'Interstellar Wars', 'Futuristic interstellar action movie. Can the rebels save
the humans from the evil empire?', 'PG', 'SCIFI', TO_DATE('07-JUL-77', 'DD-MONTH-YYYY'))

1 row created.
```

est on remarque que l'enregistrement est sauvegarder avec la duplication.

| select * from title; | | | | | |
|--|-------------------------|---|--------|----------|--------------|
| Results Explain Describe Saved SQL History | | | | | |
| TITLE_ID | TITLE | DESCRIPTION | RATING | CATEGORY | RELEASE_DATE |
| 92 | Willie and Chrismas Too | All Willie's list for Sanata, but willie has yet to add his own wish list. | G | CHILD | 05-OCT-95 |
| 93 | Alien again | yet another installation of science fiction history . can the heroine save the planet from the alien life form? | R | SCIFI | 19-MAY-95 |
| 94 | THE Glob | A meteor crashes near a small American town and unleashes carvorous goo in this classic. | NR | SCIFI | 12-AUG-95 |
| 96 | My Day Off | with a little luck and a lot of ingenuity , a teenager skips school for a day in new york. | PG | COMEDY | 12-JUL-95 |
| 97 | Miracles on ice | A six-year-old has doubts about santa claus, but she discovers that miracles really do existe. | PG | DRAMA | 12-SEP-95 |
| 99 | Soda Gang | After discovering a cache of drugs , a young couple find themselves pitted against a vicious gang. | NR | ACTION | 01-JUN-95 |
| 100 | Interstellar Wars | Futuristic interstellar action movie. Can the rebels save the humans from the evil empire? | PG | SCIFI | 07-JUL-77 |
| 101 | Interstellar Wars | Futuristic interstellar action movie. Can the rebels save the humans from the evil empire? | PG | SCIFI | 07-JUL-77 |
| 8 rows returned in 0.00 seconds CSV Export | | | | | |

- b. Entrez deux réservations : une pour Carmen Velasquez, qui souhaite louer "Interstellar Wars" et l'autre pour Mark Quick-to-See, qui souhaite louer "Soda Gang".

```
SQL> insert into reservation(res_date,number_id,title_id) values((SYSDATE+1),103,100);
1 row created.

SQL> insert into reservation(res_date,number_id,title_id) values((SYSDATE+2),107,99);
1 row created.

SQL> COMMIT;

Commit complete.
```

| | | |
|--|-----------|----------|
| select * from reservation; | | |
| | | |
| Results Explain Describe Saved SQL History | | |
| | | |
| RES_DATE | NUMBER_ID | TITLE_ID |
| 17-JUL-23 | 103 | 100 |
| 18-JUL-23 | 107 | 99 |
| 2 rows returned in 0.04 seconds | | |
| CSV Export | | |

7. Modifiez l'une des tables.

- a. Ajoutez une colonne PRICE à la table TITLE pour enregistrer le prix d'achat de la cassette vidéo. Cette colonne doit permettre la saisie de huit chiffres dont deux décimales. Vérifiez vos modifications.

| Name | Null? | Type |
|--------------|----------|---------------|
| TITLE_ID | NOT NULL | NUMBER(10) |
| TITLE | NOT NULL | VARCHAR2(60) |
| DESCRIPTION | NOT NULL | VARCHAR2(400) |
| RATING | | VARCHAR2(4) |
| CATEGORY | | VARCHAR2(20) |
| RELEASE_DATE | | DATE |
| PRICE | | NUMBER(8,2) |

Pour faire cela on exécute la requête :

```
ALTER TABLE title
```

```
ADD price number(8,2);
```

```
SQL> ALTER TABLE title ADD price number(8,2);
```

Table altered.

```
SQL> desc title;
```

| Name | Null? | Type |
|--------------|----------|---------------|
| TITLE_ID | NOT NULL | NUMBER(38) |
| TITLE | NOT NULL | VARCHAR2(60) |
| DESCRIPTION | NOT NULL | VARCHAR2(400) |
| RATING | | VARCHAR2(4) |
| CATEGORY | | VARCHAR2(20) |
| RELEASE_DATE | | DATE |
| PRICE | | NUMBER(8,2) |

b. Créez un script nommé lab14_7b.sql, contenant des instructions de mise à jour qui permettent d'ajouter à chaque cassette vidéo les prix indiqués dans la liste suivante. Exécutez les commandes du script.

| Title | Price |
|--------------------------|-------|
| Willie and Christmas Too | 25 |
| Alien Again | 35 |
| The Glob | 35 |
| My Day Off | 35 |
| Miracles on Ice | 30 |
| Soda Gang | 35 |
| Interstellar Wars | 29 |

Notre script implementra la command :

update title

set price='&price'

where title='&title';

exemple d'exécution :

```
SQL> @"C:\Users\hp\Desktop\lab14_7b.sql"
Enter value for price: 25
old 2: set price='&price'
new 2: set price='25'
Enter value for title: Willie and Chrismas Too
old 3: where title='&title'
new 3: where title='Willie and Chrismas Too'

1 row updated.
```

En exécutera le script donc pour remplir les autres champs en se termine par :

```
select * from title;
```

Results Explain Describe Saved SQL History

| TITLE_ID | TITLE | DESCRIPTION | RATING | CATEGORY | RELEASE_DATE | PRICE |
|----------|--------------------------|---|--------|----------|--------------|-------|
| 92 | Willie and Christmas Too | All Willie's list for Sanata, but willie has yet to add his own wish list. | G | CHILD | 05-OCT-95 | 25 |
| 93 | Alien again | yet another installation of science fiction history . can the heroine save the planet from the alien life form? | R | SCIFI | 19-MAY-95 | 35 |
| 94 | THE Glob | A meteor crashes near a small American town and unleashes carvorous goo in this classic. | NR | SCIFI | 12-AUG-95 | 35 |
| 96 | My Day Off | with a little luck and a lot of ingenuity , a teenager skips school for a day in new york. | PG | COMEDY | 12-JUL-95 | 35 |
| 97 | Miracles on ice | A six-year-old has doubts about santa claus, but she discovers that miracles really do existe. | PG | DRAMA | 12-SEP-95 | 30 |
| 99 | Soda Gang | After discovering a cache of drugs , a young couple find themselves pitted against a vicious gang. | NR | ACTION | 01-JUN-95 | 35 |
| 100 | Interstellar Wars | Futuristic interstellar action movie. Can the rebels save the humans from the evil empire? | PG | SCIFI | 07-JUL-77 | 29 |
| 101 | Interstellar Wars | Futuristic interstellar action movie. Can the rebels save the humans from the evil empire? | PG | SCIFI | 07-JUL-77 | 29 |

8 rows returned in 0.00 seconds

[CSV Export](#)