MODULE III :

DML (Data Manipulation Language - Langage de manipulation de données)

Exercice 3.1 – **Inscrivez-vous comme étudiant dans la base de données DBSlide sans spécifier les noms de colonnes dans lesquelles on insère les données**

```
INSERT INTO student
 VALUES (26, 'Will', 'Smith', '1968-09-25', 'wsmith', 1010, 13, 'EG2210')
```

Exercice 3.2 – **Inscrivez votre voisin comme étudiant dans la base de données DBSlide. Votre voisin n'aura ni nom de famille, ni login, ni résultat annuel (valeurs NULL)**

```
INSERT INTO student (student_id, first_name, birth_date, section_id, course_id)
 VALUES (27, 'Russel', '1964-04-07', 1010, 'EG2210')
```

Exercice 3.3 – **Créer une table « section_archives » qui contiendra une copie des données contenues dans la table section**

```
CREATE TABLE section_archives (
    section_id_arch int primary key,
    section_name_arch varchar(50),
    delegate_id_arch int
)

INSERT INTO section_archives
SELECT * FROM section
```

Exercice 3.4 – **Insérer un nouvel étudiant dans la base de données. Cet étudiant sera inscrit dans la même section que Keanu Reeves, assistera au cours donné par le professeur Zidda (les lettres 'EG' suivies des 4 derniers caractères du cours en question) mais n'aura pas de login (données que la requête devra retrouver, sans que vous les renseigniez directement)**

```
INSERT INTO student (student_id, first_name, last_name, birth_date,
                     section_id, year_result, course_id)
VALUES (28, 'Chuck', 'Norris', '1940-03-10',
        (SELECT section_id FROM student WHERE last_name like 'Reeves'),
        8, 'EG' + RIGHT((SELECT course_id
                         FROM course c1 JOIN professor p
                              ON c1.professor_id = p.professor_id
                         WHERE p.professor_name like 'Zidda'),4))
```

```
INSERT INTO student (student_id, first_name, last_name, birth_date,
                     section_id, year_result, course_id)
SELECT 28, 'Chuck', 'Norris', '1940-03-10',
       s.section_id, 8, 'EG' + RIGHT(c.course_id,4)
FROM section s, course c
WHERE s.section_id = (SELECT section_id FROM student
                      WHERE last_name like 'Reeves')
      AND c.course_id = (SELECT course_id
                         FROM course c1 JOIN professor p
                              ON c1.professor_id = p.professor_id
                         WHERE p.professor_name like 'Zidda')
```

Exercice 3.5 – **Insérer une nouvelle section dans la table section qui portera l'ID de section 1530, qui aura l'intitulé « Administration des SI » et qui aura le même délégué que la section dont l'ID et 1010 (vous ne connaissez pas la valeur de l'ID de ce délégué)**

```
INSERT INTO section
SELECT 1530, 'Administration des SI', delegate_id
FROM section WHERE section_id = 1010
```

Exercice 3.6 – **Mettre à jour vos propres données pour vous inscrire au cours EG2210**

```
UPDATE student
 SET course_id = 'EG2210'
 WHERE student_id = 26
```

Exercice 3.7 – **Mettre à jour les données de votre voisin pour qu'il ait un nom. Ensuite, refaire une mise à jour de la même ligne de données et attribuer à votre voisin un résultat de 18/20 et un login correspondant à la concaténation de la première lettre de son prénom et de la totalité de son nom, le tout en minuscules (sans connaître les valeurs réelles du nom et du prénom utilisés)**

```
UPDATE student
SET year_result = '18',
    last_name   = 'Crowe',
    login       = LOWER(LEFT((SELECT first_name FROM student WHERE student_id=27),1)
                  + '' +
                  (SELECT last_name FROM student WHERE student_id=27))
WHERE student_id = 27;
```

Exercice 3.8 – **Mettre à jour les données de la table « student » pour que tous les étudiants de la section 1010 aient 15/20**

```
UPDATE student
 SET year_result = 15
 WHERE section_id = 1010

```

Exercice 3.9 – **Nommer Keanu Reeves délégué de la section 1530 (sans connaître la valeur réelle de l'ID de M. Reeves)**

```
UPDATE section
SET delegate_id = (SELECT student_id
                   FROM student
                   WHERE first_name = 'Keanu'
                   AND last_name = 'Reeves')
WHERE section_id = 1530
```

Exercice 3.10 – **Donner à la section 1530 le même nom de section et le même délégué que la section 1320 (en allant rechercher ces valeurs via la requête, pas en les renseignant directement)**

```
UPDATE section
 SET delegate_id = (SELECT delegate_id
                    FROM section
                    WHERE section_id = '1320'
                   ),
     section_name =(SELECT section_name
                    FROM section
                    WHERE section_id = '1320'
                   )
 WHERE section_id = 1530
```

Exercice 3.11 – **Nommer Alyssa Milano déléguée de sa section. On ne connait pas la valeur réelle de la section dans laquelle Mlle Milano est inscrite**

```
UPDATE section
 SET section.delegate_id = student.student_id
 FROM section JOIN student
      ON section.section_id = student.section_id
 WHERE student.last_name LIKE 'Milano'
```

Exercice 3.12 – **Supprimer votre voisin de la base de données**

```
1  DELETE FROM student WHERE student_id=27
2
```

Exercice 3.13 – **Retirez-vous ainsi que Kim Basinger de la base de données. Comment se fait-il que le système accepte cette manipulation alors que Mlle. Basinger est déléguée de section ?**

Solution : la colonne delegate_id n'est pas une clé étrangère.

```
1  DELETE FROM student
2   WHERE student_id=26
3   OR (first_name = 'Kim' AND last_name = 'Basinger')
```

Exercice 3.14 – **Supprimer tous les étudiants qui ont moins de 8/20**

```
1  DELETE FROM student
2   WHERE year_result < 8
```

Exercice 3.15 – **Supprimer tous les cours qui n'ont pas de professeur**

```
1  DELETE FROM course
2   WHERE professor_id NOT IN (SELECT professor_id
3                              FROM professor)
```

<u>Exercice 3.16 (bonus DDL-DML)</u> – Sans supprimer les clés étrangères au préalable, supprimer les données de toutes les tables dans l'ordre suivant : sections => professeurs => étudiants => cours => grades. Il est possible qu'il faille d'abord modifier la structure des tables (ALTER TABLE) afin d'accepter des valeurs nulles à certains endroits… Une modification des données des tables sous-jacentes avant la suppression de certaines données sera sûrement nécessaire également

```
1  BEGIN TRANSACTION emptyTables
2  DECLARE @errors INT
3  SET @errors = 0
4
5  ALTER TABLE student NOCHECK CONSTRAINT FK_student_section
6  ALTER TABLE course NOCHECK CONSTRAINT FK_course_professor
7  ALTER TABLE professor NOCHECK CONSTRAINT FK_professor_section
8
9  DELETE FROM section
10 SET @errors= @errors+ @@ERROR
11
12 DELETE FROM professor
13 SET @errors= @errors+ @@ERROR
14
15 DELETE FROM student
16 SET @errors= @errors+ @@ERROR
17
18 DELETE FROM course
19 SET @errors= @errors+ @@ERROR
20
21 DELETE FROM grade
22 SET @errors= @errors+ @@ERROR
23
24 ALTER TABLE student WITH CHECK CHECK CONSTRAINT FK_student_section
25 ALTER TABLE course WITH CHECK CHECK CONSTRAINT FK_course_professor
26 ALTER TABLE professor WITH CHECK CHECK CONSTRAINT FK_professor_section
27
28 PRINT 'Erreur: ' + CAST(@errors AS VARCHAR(10))
29
30 IF @errors = 0
31      COMMIT TRANSACTION emptyTables
32 ELSE
33      ROLLBACK TRANSACTION emptyTables
34
```