

SQL Déclaratif

Du standard à la pratique

Manuel d'exercices

MODULE I :

DDL (Data Definition Language - Langage de définition de données)

Exercice 1.1 – La syntaxe des ordres suivants est-elle correcte ? Si non, pourquoi ?

Attention, les tables sont peut-être liées... !

N'hésitez pas à tester les requêtes directement !

```
1 CREATE TABLE T_office
2 ( office_id INTEGER,
3 office_address VARCHAR(30),
4 CONSTRAINT PK_office PRIMARY KEY (office_id))
5
6 CREATE TABLE T_course
7 ( crs_code CHAR(8) NOT NULL PRIMARY KEY,
8 crs_name VARCHAR(30),
9 CONSTRAINT UK_crs UNIQUE (crs_name))
10
11 CREATE TABLE T_professor
12 ( prf_id INTEGER NOT NULL PRIMARY KEY,
13 prf_name VARCHAR(30),
14 prf_course CHAR(8), FOREIGN KEY (prf_course)
15 CONSTRAINT PK_course REFERENCES T_course (crs_code)
16 ON DELETE SET NULL,
17 office_id CHAR(2) REFERENCES T_office (office_id)
18 CONSTRAINT prf_name UNIQUE (prf_name))
```

Exercice 1.2 – A partir des données présentées dans le tableau suivant, proposer le code de la table T_MAINTENANCE_MTN.

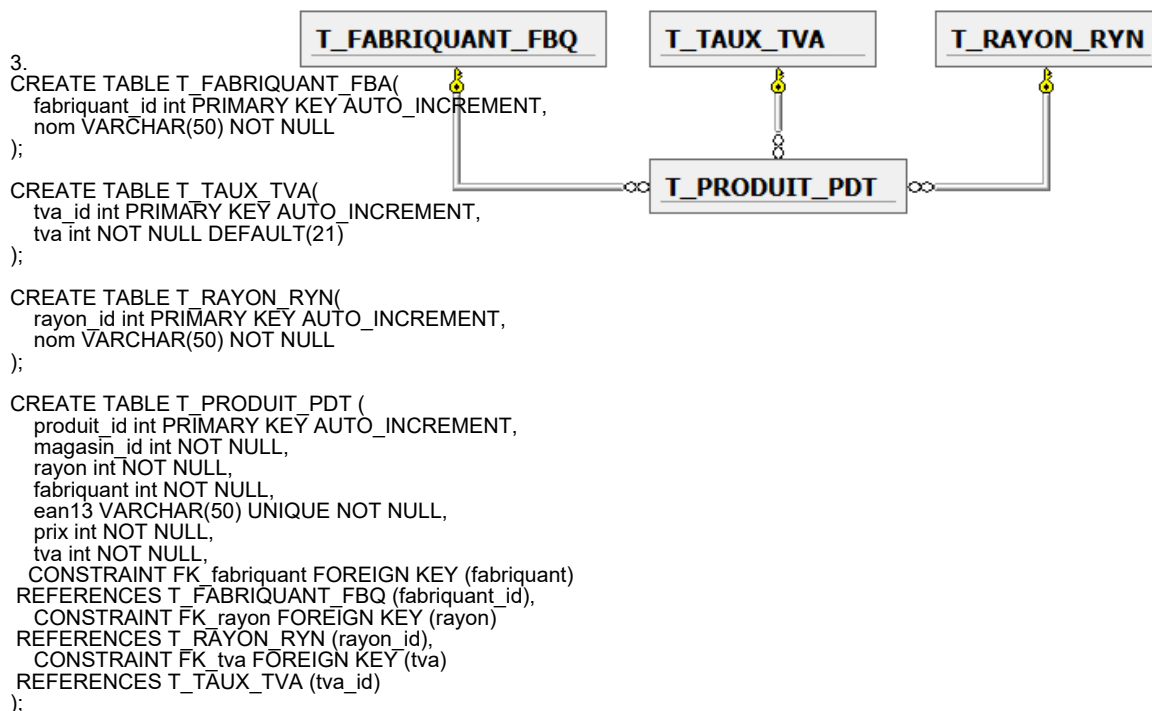
Cette table devra contenir les 4 contraintes suivantes : contrainte de clé primaire, contrainte d'unicité, contrainte check et contrainte NOT NULL. Ces contraintes porteront sur 4 colonnes ou combinaisons de colonnes distinctes.

Jour	Machine	Numéro	Vitesse	Température	Heure	Evénement
Ven	Massicot	147			21 :18	Défaut de lame
Sam	Relieuse	63	16		16 :15	Arrêt pour maintenance
Jeu	Presse	87	6	62	11 :40	Bavure encre
Sam	Relieuse	79	16		17 :11	Reprise
Mer	Presse	89	6	55	08 :28	Recadrage
Mar	Presse	132	8	68	09 :58	Changement encre
Mer	Massicot	111			10 :17	Graissage coulisseau

```
CREATE TABLE T_MAINTENANCE_MTN(
  mtm_id int PRIMARY KEY AUTO_INCREMENT,
  jour CHAR(3) NOT NULL CHECK (jour IN ("Lun", "Mar", "Mer", "Jeu", "Ven", "Sam", "Dim")),
  machine VARCHAR(20) NOT NULL,
  numero int NOT NULL,
  vitesse int CHECK (vitesse>0),
  temperature int,
  heure datetime UNIQUE,
  evenement VARCHAR(50) NOT NULL,
);
```

Exercice 1.3 – Créer une table pour y stocker les produits à vendre, avec les rubriques suivantes : identifiant, référence magasin, référence fabricant, code EAN13, prix de vente. Cette table fera en outre référence aux tables T_TAUX_TVA, T_RAYON_RYN, T_FABRICANT_FBQ.

Mettez en place toutes les contraintes nécessaires. La table produit contiendra au minimum les colonnes proposées, mais peut en contenir d'autres au besoin.



Exercice 1.4 – Soit le code de création de table repris ci-après et pour lequel les annotations suivantes concernant les fonctions utilisées pourront être utiles (sous Oracle, demander le script et des explications au formateur) :

- « RTRIM(...) » et « LTRIM(...) » enlève les espaces blancs respectivement à droite et à gauche de l'élément entre parenthèses
- « SUBSTRING(...,x,y) » renvoi la chaîne de caractère commençant à « x » et se terminant « y » caractères après « x », à partir de la chaîne de caractères donnée entre parenthèses
- « CONVERT(TYPE,...) » renvoi la valeur fournie dans le « TYPE » demandé

```

1 CREATE TABLE T_VOITURE_VTR (
2     VTR_ID          INTEGER      NOT NULL PRIMARY KEY,
3     VTR_IMMATRICUL  CHAR(10)     NOT NULL UNIQUE,
4     VTR_CARBURANT    CHAR(2)     NOT NULL DEFAULT 'ES' CHECK(VTR_CARBURANT IN ('ES', 'GO', 'PL')),
5     VTR_PUISSANCE_FISC INTEGER    NOT NULL CHECK(VTR_PUISSANCE_FISC BETWEEN 1 AND 20),
6     VTR_NB_PLACES    INTEGER     NOT NULL CHECK(VTR_NB_PLACES BETWEEN 1 AND 7),
7     VTR_MODELE       VARCHAR(20) CHECK(RTRIM(LTRIM(VTR_MODELE)) NOT LIKE ''),
8     VTR_CONSTRUCTEUR VARCHAR(16) CHECK(RTRIM(LTRIM(VTR_CONSTRUCTEUR)) NOT LIKE ''),
9     VTR_NUMERO_SERIE VARCHAR(25) NOT NULL CHECK(RTRIM(LTRIM(VTR_NUMERO_SERIE)) NOT LIKE ''),
10    CONSTRAINT CK_IMMATRICULATION CHECK(((CONVERT(INTEGER, SUBSTRING(VTR_IMMATRICUL, 9, 1)) BETWEEN 0 AND 9)
11        AND (SUBSTRING(VTR_IMMATRICUL, 10, 1) BETWEEN '0' AND '9')
12        AND (SUBSTRING(VTR_IMMATRICUL, 9, 2) < '96'))
13        OR ((CONVERT(INTEGER, SUBSTRING(VTR_IMMATRICUL, 9, 1)) = 2)
14        AND (SUBSTRING(VTR_IMMATRICUL, 10, 1) IN ('A', 'B')))),
15    CONSTRAINT CK_PUISS_PLACE CHECK(VTR_NB_PLACES - 1 < VTR_PUISSANCE_FISC),
16    CONSTRAINT UK_MDL_CTR_NSR UNIQUE (VTR_MODELE, VTR_CONSTRUCTEUR, VTR_NUMERO_SERIE)
17 -)

```

Parmi les lignes suivantes, lesquelles seront refusées et pourquoi ?

ID	IMMA	CARB	PUISS	PLC	MDL	CONST	NUM_SERIE
14	'478 XDA 78'	'ES'	9	5	'305'	'PEUGEOT'	'00014578'
31	'1447 MD 44'	'ES'	'7'	5		'CITROEN'	'0001578'
7	'5475 MRT 91'	'GO'	5	4	'204'	'PEUGEOT'	'0001474578'
11	'1744 BC 76'	'GO'	7	5			'00025678'
15	'4412 LR 75'	'GO'	7	4	'305'	'PEUGEOT'	'00014578'
17	'971 VTR 96'		7	5	'306'	'PEUGEOT'	'00017548'
19	'991 SDT 75'	'ES'	8	5	'MEGANE'	'RENAULT'	'00014578'
20	'991 SDT 75'	'ES'	5	4	'MEGANE'	'RENAULT'	'00014578'
14	'4875 ZT 94'		7	5		'RENAULT'	'005784'
7	'5474 MRT 91'	'GPL'	5	4	'PT CRUISER'	'CHRYSLER'	'0000050214'

Exercice 1.5 – Deux scripts vous sont fournis : « DBSlide_LoadDB.sql » et « DBSlide_LoadData.sql ».

Créer une base de données que l'on appellera « DBSlide ». Tenter d'exécuter les scripts fournis... Cela ne devrait pas fonctionner. A vous de les corriger !

Exercice 1.6 – Une fois les scripts de l'exercice précédent corrigés, les tables créées et remplies, réaliser les modifications suivantes :

- Autoriser la table « SECTION » à accepter des valeurs NULL pour la colonne « delegate_id »
- Ajouter à la table « SECTION » une clé étrangère faisant pointer la colonne « delegate_id » vers la colonne « student_id » de la table « STUDENT »
- Supprimer la colonne « course_id » de la table « STUDENT »
- Faire en sorte que les données de la colonne « student_id » de la table « STUDENT » soient auto-incrémentées
- En ne supprimant aucune donnée, modifier le type de la colonne « section_id » de la table « section » afin qu'il soit en CHAR(4). Cela impliquera peut-être d'autres modifications...

Exercice 1.7 – Améliorer le script « DBSlide_LoadDB.sql » afin qu'il commence par supprimer les tables, pour ensuite les recréer sans leurs clés étrangères. Une fois chaque table créée, leur rajouter les clés étrangères

Exercice 1.8 – Afin de partir sur des bases communes pour les exercices à venir, exécuter les scripts « DBSlide_LoadDB_OK.sql » et « DBSlide_loadData_OK.sql » (ou « DBSlide_LoadDB_Oracle.sql » et « DBSlide_LoadData_Oracle.sql », sous Oracle)

Exercice 1.9 – Ceci clôture la partie DDL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)

MODULE 2 :

DRL (Data Retrieval Language - Langage d'extraction de données)

Partie I : SELECT ... FROM ...

Exercice 2.1.1 – Les requêtes suivantes fonctionnent-elles sous SQL-Server ? Si non, comment les corriger ?

N'hésitez pas à tester vos requêtes directement sous Management Studio !

```
1 SELECT last_name, first_name AS "F name"
2 FROM student;
3
4 SELECT last_name lname, first_name AS fname
5 FROM student;
6     CONCAT(last_name, " ", first_name)
7 SELECT last_name || ' ' || first_name AS name
8 FROM student;
9     CONCAT(last_name, " ", first_name)
10 SELECT last_name+first_name AS name, Year_result * 10 result,
11 FROM student;
```

Exercice 2.1.2 – Ecrire une requête pour présenter, pour chaque étudiant, le nom de l'étudiant, la date de naissance, le login et le résultat pour l'année de l'ensemble des étudiants.

```
SELECT last_name AS "Nom",
birth_date AS "Date de naissance",
login AS "Login",
year_result AS "Résultat annuele"
FROM student;
SELECT
AVG(year_result) AS "Moyenne générale"
FROM student;
```

Exercice 2.1.3 – Ecrire une requête pour présenter, pour chaque étudiant, son nom complet (nom et prénom séparés par un espace), son id et sa date de naissance.

```
SELECT
CONCAT(last_name, " ", first_name) AS name,
student_id id,
birth_date as "Date de naissance"
FROM student;
```

Exercice 2.1.4 – Ecrire une requête pour présenter, pour chaque étudiant, dans une seule colonne (nommée « Info Étudiant ») l'ensemble des données relatives à un étudiant séparées par le symbole « | ». Sous SQL Server, il est nécessaire d'avoir recours à la fonction de conversion CONVERT(type, champs).

```
SELECT
CONCAT(student_id, " | ", last_name, " | ", first_name, " | ", birth_date, " | ", login, " | ", section_id, " | ", year_result, " | ", course_id) AS "Info Étudiant"
FROM student;
```

Partie II : SELECT ... FROM ... WHERE ... ORDER BY

Exercice 2.2.1 – Ecrire une requête pour présenter le login et le résultat de tous les étudiants ayant obtenu un résultat annuel supérieur à 16

	login	year_result
1	kbasinge	19
2	jroberts	17
3	agarcia	19
4	jgarner	18
5	hberry	18

```
SELECT
  login,
  year_result
FROM student
WHERE year_result > 16;
```

Exercice 2.2.2 – Ecrire une requête pour présenter le nom et l'id de section des étudiants dont le prénom est Georges

	last_name	section_id
1	Lucas	1320
2	Clooney	1020

```
SELECT
  last_name,
  section_id
FROM student
WHERE first_name IN ("Georges");
```

Exercice 2.2.3 – Ecrire une requête pour présenter le nom et le résultat annuel de tous les étudiants ayant obtenu un résultat annuel compris entre 12 et 16

	last_name	year_result
1	Connery	12
2	Bacon	16

```
SELECT
  last_name,
  year_result
FROM student
WHERE year_result >= 12 AND year_result <= 16;
```

Exercice 2.2.4 – Ecrire une requête pour présenter le nom, l'id de section et le résultat annuel de tous les étudiants qui ne font pas partie des sections 1010, 1020 et 1110

	last_name	section_id	year_result
1	Lucas	1320	10
2	Bacon	1120	16
3	Basinger	1310	19
4	Roberts	1120	17
5	Garner	1120	18
6	Fox	1310	3
7	Doherty	1320	2
8	Berry	1320	18

```
SELECT
  last_name,
  section_id,
  year_result
FROM student
WHERE section_id NOT IN ("1010", "1020", "1110");
```

Exercice 2.2.5 – Ecrire une requête pour présenter le nom et l'id de section de tous les étudiants qui ont un nom de famille qui termine par « r »

	last_name	section_id
1	Basinger	1310
2	Michelle Gellar	1020
3	Garner	1120

```

SELECT
  last_name,
  section_id
FROM student
WHERE last_name LIKE "%_r";

```

Exercice 2.2.6 – Ecrire une requête pour présenter le nom et le résultat annuel de tous les étudiants qui ont un nom de famille pour lequel la troisième lettre est un « n » et qui ont obtenu un résultat annuel supérieur à 10

	last_name	year_result
1	Connery	12

```

SELECT
  last_name,
  year_result
FROM student
WHERE last_name LIKE "__n%" AND year_result > 10;

```

Exercice 2.2.7 – Ecrire une requête pour présenter le nom et le résultat annuel classé par résultats annuels décroissants de tous les étudiants qui ont obtenu un résultat annuel inférieur ou égal à 3

	last_name	year_result
1	De Niro	3
2	Fox	3
3	Morse	2
4	Bullock	2
5	Doherty	2

```

SELECT
  last_name,
  year_result
FROM student
WHERE year_result <= 3
ORDER BY year_result DESC;

```

Exercice 2.2.8 – Ecrire une requête pour présenter le nom complet (nom et prénom séparés par un espace) et le résultat annuel classé par nom croissant sur le nom de tous les étudiants appartenant à la section 1010

	Nom complet	year_result
1	Bullock Sandra	2
2	Eastwood Clint	4
3	Portman Natalie	4
4	Willis Bruce	6

```

SELECT
  CONCAT(last_name, " ", first_name) AS "Nom complet",
  year_result
FROM student
WHERE section_id IN ("1010")
ORDER BY `Nom complet` ASC;

```

Exercice 2.2.9 – Ecrire une requête pour présenter le nom, l'id de section et le résultat annuel classé par ordre croissant sur la section de tous les étudiants appartenant aux sections 1010 et 1020 ayant un résultat annuel qui n'est pas compris entre 12 et 18

	last_name	section_id	year_result
1	Willis	1010	6
2	Eastwood	1010	4
3	Portman	1010	4
4	Bullock	1010	2
5	Reeves	1020	10
6	Clooney	1020	4
7	Cruise	1020	4
8	Witherspoon	1020	7
9	Michelle Gellar	1020	7
10	Hanks	1020	8

```
SELECT
  last_name,
  section_id,
  year_result
FROM student
WHERE section_id IN (1010, 1020) AND year_result NOT BETWEEN 12 AND 18
ORDER BY section_id ASC;
```

Exercice 2.2.10 – Ecrire une requête pour présenter le nom, l'id de section et le résultat annuel sur 100 (nommer la colonne « Résultat sur 100 ») classé par ordre décroissant du résultat de tous les étudiants appartenant aux sections commençant par 13 et ayant un résultat annuel sur 100 inférieur ou égal à 60

	last_name	section_id	Résultat sur 100
1	Lucas	1320	50
2	Fox	1310	15
3	Doherty	1320	10

```
SELECT
  last_name,
  section_id,
  (year_result*100/20) AS "Résultat sur 100"
FROM student
WHERE section_id LIKE "13%" AND (year_result*100/20) <= 60
ORDER BY year_result DESC;
```

Exercice 2.2.11 – Ceci clôture la première partie DRL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)

Partie III : Les Fonctions

Exercice 2.3.1 – Pourquoi lorsque l'on utilise la fonction « MAX » ou « MIN » les valeurs « NULL » sont-elles ignorées ?

PARCE QUE

Exercice 2.3.2 – Pourquoi le type des données n'a-t-il pas d'importance lorsque l'on utilise la fonction « COUNT » ?

Parce que COUNT compte la quantité de lignes d'un tableau

Exercice 2.3.3 – La fonction « AVG » renvoie la moyenne de toutes les lignes résultantes d'une requête SELECT sur une colonne incluant toutes les valeurs « NULL ». (Vrai/Faux ?)

FAUX, il ignore les valeurs NULL

Exercice 2.3.4 – La fonction « SUM » est utilisée pour ajouter des totaux aux colonnes. (Vrai/Faux ?)

FAUX, c'est pour avoir un total de tous les valeurs d'une colonne.

Exercice 2.3.5 – La fonction « COUNT(*) » compte toutes les lignes d'une table. (Vrai/Faux ?)

VRAI

Exercice 2.3.6 – Les requêtes suivantes sont-elles valides ?

```
1 SELECT COUNT *  
2 FROM student;  
3  
4 SELECT COUNT(student_id), login  
5 FROM student;  
6  
7 SELECT MIN(year_result), MAX(birth date)  
8 FROM student  
9 WHERE year_result > 12;
```

Exercice 2.3.7 – Donner le résultat annuel moyen pour l'ensemble des étudiants

```
SELECT
  AVG(year_result)
FROM student;
```

Exercice 2.3.8 – Donner le plus haut résultat annuel obtenu par un étudiant

```
SELECT
  MAX(year_result)
FROM student;
```

Exercice 2.3.9 – Donner la somme des résultats annuels

```
SELECT
  SUM(year_result)
FROM student;
```

Exercice 2.3.10 – Donner le résultat annuel le plus faible

```
SELECT
  MIN(year_result)
FROM student;
```

Exercice 2.3.11 – Donner le nombre de lignes qui composent la table « STUDENT »

```
SELECT
  COUNT(*)
FROM student;
```

Exercice 2.3.12 – Donner la liste des étudiants (login et année de naissance) nés après 1970

	login	Année de naissance
1	nportman	1981
2	rwithers	1976
3	smichell	1977
4	amilano	1972
5	jgarner	1972
6	sdoherty	1971

```
SELECT
  login,
  DATE_FORMAT(birth_date, "%Y") AS "Année de naissance"
FROM student
WHERE DATE_FORMAT(birth_date, "%Y") > 1970
```

Exercice 2.3.13 – Donner le login et le nom de tous les étudiants qui ont un nom composé d'au moins 8 lettres

	login	last_name
1	ceastwoo	Eastwood
2	kbasinge	Basinger
3	rwithers	Witherspoon
4	smichell	Michelle Gellar

```
SELECT
  login,
  last_name
FROM student
WHERE CHAR_LENGTH(last_name) >= 8;
```

Exercice 2.3.14 – Donner la liste des étudiants ayant obtenu un résultat annuel supérieur ou égal à 16. La liste présente le nom de l'étudiant en majuscules (nommer la colonne « Nom de Famille ») et le prénom de l'étudiant dans l'ordre décroissant des résultats annuels obtenus

	Nom de famille	first_name	year_result
1	BASINGER	Kim	19
2	GARCIA	Andy	19
3	GARNER	Jennifer	18
4	BERRY	Halle	18
5	ROBERTS	Julia	17
6	BACON	Kevin	16

```
SELECT
  UPPER(last_name) AS "Nom de famille",
  first_name,
  year_result
FROM student
WHERE year_result >= 16
ORDER BY year_result DESC;
```

Exercice 2.3.15 – Donner un nouveau login à chacun des étudiants ayant obtenu un résultat annuel compris entre 6 et 10. Le login se compose des deux premières lettres du prénom de l'étudiant suivi par les quatre premières lettres de son nom le tout en minuscule. Le résultat reprend pour chaque étudiant, son nom, son prénom l'ancien et le nouveau login (colonne « Nouveau login »)

	first_name	last_name	login	Nouveau login
1	Georges	Lucas	glucas	geluca
2	Bruce	Willis	bwillis	brwill
3	Reese	Witherspoon	rwithers	rewith
4	Sophie	Marceau	smarceau	somarc
5	Sarah	Michelle Gellar	smichell	samich
6	Alyssa	Milano	amilano	almila
7	Tom	Hanks	thanks	tohank
8	Keanu	Reeves	kreeves	kereev

```
SELECT
  first_name,
  last_name,
  login,
  LOWER(CONCAT(LEFT(first_name, 2), LEFT(last_name, 4))) AS "Nouveau login"
FROM student;
```

Exercice 2.3.16 – Donner un nouveau login à chacun des étudiants ayant obtenu un résultat annuel égal à 10, 12 ou 14. Le login se compose des trois dernières lettres de son prénom suivi du chiffre obtenu en faisant la différence entre l'année en cours et l'année de leur naissance. Le résultat reprend pour chaque étudiant, son nom, son prénom l'ancien et le nouveau login (colonne « Nouveau login »)

	first_name	last_name	login	Nouveau login
1	Georges	Lucas	glucas	ges69
2	Sean	Connery	sconnery	ean83
3	Keanu	Reeves	kreeves	anu49

```
SELECT
  first_name,
  last_name,
  login,
  LOWER(CONCAT(RIGHT(first_name, 3), YEAR(CURDATE())-YEAR(birth_date))) AS "Nouveau login"
FROM student
WHERE year_result IN ("10", "12", "14");
```

Exercice 2.3.17 – Donner la liste des étudiants (nom, login, résultat annuel) qui ont un nom commençant par « D », « M » ou « S ». La liste doit présenter les données dans l'ordre croissant des dates de naissance des étudiants

```
SELECT
  last_name,
  login,
  year_result
FROM student
WHERE LEFT(last_name, 1) IN ("D", "M", "S")
ORDER BY birth_date ASC;
```

	last_name	login	year_result
1	De Niro	rde niro	3
2	Morse	dmorse	2
3	Depp	jdepp	11
4	Marceau	smarceau	6
5	Doherty	sdoherty	2
6	Milano	amilano	7
7	Michelle Gellar	smichell	7

Exercice 2.3.18 – Donner la liste des étudiants (nom, login, résultat annuel) qui ont obtenu un résultat impair supérieur à 10. La liste doit être triée du plus grand résultat au plus petit

```
SELECT
  last_name,
  login,
  year_result
FROM student
WHERE year_result%2 > 0 AND year_result > 10
ORDER BY year_result DESC;
```

	last_name	login	year_result
1	Basinger	kbasinge	19
2	Garcia	agarcia	19
3	Roberts	jroberts	17
4	Depp	jdepp	11

Exercice 2.3.19 – Donner le nombre d'étudiants qui ont au moins 7 lettres dans leur nom de famille

```
SELECT
  COUNT(*) AS "Nbre de noms de plus de 7 lettres"
FROM student
WHERE CHAR_LENGTH(last_name) >= 7
ORDER BY year_result DESC;
```

	Nbre de noms de plus de 7 lettres
1	12

Exercice 2.3.20 – Pour chaque étudiant né avant 1955, donner le nom, le résultat annuel et le statut. Le statut prend la valeur « OK » si l'étudiant a obtenu au moins 12 comme résultat annuel et « KO » dans le cas contraire

```
SELECT
  last_name,
  year_result,
  CASE
    WHEN year_result >= 12 THEN "OK"
    ELSE "KO"
  END AS "Statu"
FROM student
WHERE YEAR(birth_date) < 1955;
```

	last_name	year_result	Statut
1	Lucas	10	KO
2	Eastwood	4	KO
3	Connery	12	OK
4	De Niro	3	KO
5	Basinger	19	OK
6	Morse	2	KO

Exercice 2.3.21 – Donner pour chaque étudiant né entre 1955 et 1965 le nom, le résultat annuel et la catégorie à laquelle il appartient. La catégorie est fonction du résultat annuel obtenu : un résultat inférieur à 10 appartient à la catégorie « inférieure », un résultat égal à 10 appartient à la catégorie « neutre », un résultat autre appartient à la catégorie « supérieure »

	last_name	year_result	Catégorie
1	Bacon	16	superieure
2	Depp	11	superieure
3	Clooney	4	inferieure
4	Garcia	19	superieure
5	Willis	6	inferieure
6	Cruise	4	inferieure
7	Hanks	8	inferieure
8	Bullock	2	inferieure
9	Reeves	10	neutre

```

SELECT
  last_name,
  year_result,
  CASE
    WHEN year_result < 10 THEN "inferieure"
    WHEN year_result = 10 THEN "neutre"
    ELSE "superieure"
  END AS "Catégorie"
FROM student
WHERE YEAR(birth_date) BETWEEN 1955 AND 1965;

```

Exercice 2.3.22 – Donner pour chaque étudiant né entre 1975 et 1985, son nom, son résultat annuel et sa date de naissance sous la forme: jours en chiffre, mois en lettre et années en quatre chiffres (ex : 11 juin 2005)

	last_name	year_result	Literal_date
1	Portman	4	9 juin 1981
2	Witherspoon	7	22 mars 1976
3	Michelle Gellar	7	14 avril 1977

```

SELECT
  last_name,
  year_result,
  DATE_FORMAT(birth_date, "%d %M %Y")
FROM student
WHERE YEAR(birth_date) BETWEEN 1975 AND 1985;

```

Exercice 2.3.23 – Donner pour chaque étudiant né en dehors des mois d’hiver et ayant obtenu un résultat inférieur à 7, son nom, le mois de sa naissance (en chiffre) son résultat annuel et son résultat annuel corrigé (« Nouveau résultat ») tel que si le résultat annuel est égal à 4, le valeur proposée est « NULL »

	last_name	Mois de naissance	year_result	Nouveau résultat
1	Eastwood	5	4	NULL
2	De Niro	8	3	3
3	Portman	6	4	NULL
4	Clooney	5	4	NULL
5	Cruise	7	4	NULL
6	Marceau	11	6	6
7	Fox	6	3	3
8	Morse	10	2	2
9	Bullock	7	2	2
10	Doherty	4	2	2

```

SELECT
  last_name,
  MONTH(birth_date) as "Mois de naissance",
  year_result,
  NULLIF(year_result, 4) as "Nouveau résultat"
FROM student
WHERE year_result < 7 AND MONTH(birth_date) NOT IN (12, 1, 2, 3);

```

Exercice 2.3.24 – Ceci clôture la deuxième partie DRL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)

Partie IV : GROUP BY ... HAVING

Exercice 2.4.1 – L'utilisation de « GROUP BY » peut être considérée comme une forme de boucle dans une requête SQL ? (Vrai/Faux)

Exercice 2.4.2 – La répartition en groupe se fait avant de prendre en compte les restrictions imposées par un « WHERE » ? (Vrai/Faux)


Exercice 2.4.3 – Un « GROUP BY » doit impérativement porter sur une colonne non alliée ?

Exercice 2.4.4 – L'utilisation d'un « GROUP BY » a pour effet de trier les résultats dans l'ordre croissant de la colonne incluse dans le « GROUP BY » ? (Vrai/Faux)

Exercice 2.4.5 – La colonne sur laquelle porte le « GROUP BY » doit impérativement être présente dans la clause « SELECT » ? (Vrai/Faux)

Exercice 2.4.6 – Les requêtes suivantes sont-elles valides ?

```
1 SELECT section_id, count(last_name)
2 FROM student
3 GROUP BY last_name, section_id;
4
5 SELECT section_id, AVG(year_result)
6 FROM student
7 WHERE AVG(year_result) > 50
8 GROUP BY section_id;
9 HAVING AVG(year_result) > 50;
10 SELECT section_id, AVG(year_result)
11 FROM student
12 WHERE year_result > 10
13 GROUP BY section_id;
```



Exercice 2.4.7 – Donner pour chaque section, le résultat maximum (dans une colonne appelée « Résultat maximum ») obtenu par les étudiants

	section_id	Résultat maximum
1	1010	6
2	1020	12
3	1110	19
4	1120	18
5	1310	19
6	1320	18

```
SELECT
  section_id,
  MAX(year_result) as "Résultat maximum"
FROM student
GROUP BY section_id;
```

Exercice 2.4.8 – Donner pour toutes les sections commençant par 10, le résultat annuel moyen PRÉCIS (dans une colonne appelée « Moyenne ») obtenu par les étudiants

	section_id	Moyenne
1	1010	4
2	1020	7,42857142857143

```
SELECT
  section_id,
  AVG(year_result) as "Moyenne"
FROM student
GROUP BY section_id
HAVING section_id LIKE "10%";
```

Exercice 2.4.9 – Donner le résultat moyen (dans une colonne appelée « Moyenne ») et le mois en chiffre (dans une colonne appelée « Mois de naissance ») pour les étudiants nés le même mois entre 1970 et 1985

	Mois de naissance	Moyenne
1	3	7
2	4	9
3	6	4
4	12	7

```
SELECT
  MONTH(birth_date) as "Mois de naissance",
  AVG(year_result) as "Moyenne"
FROM student
WHERE YEAR(birth_date) BETWEEN 1970 AND 1985
GROUP BY "Mois de naissance";
```

Exercice 2.4.10 – Donner pour toutes les sections qui comptent plus de 3 étudiants, la moyenne PRÉCISE des résultats annuels (dans une colonne appelée « Moyenne »)

	section_id	Moyenne
1	1010	4
2	1020	7,42857142857143
3	1110	8

```
SELECT
  section_id,
  AVG(year_result) as "Moyenne"
FROM student
GROUP BY section_id
HAVING COUNT(section_id) > 3;
```


Exercice 2.4.11 – Donner le résultat maximum obtenu par les étudiants appartenant aux sections dont le résultat moyen est supérieur à 8

	section_id	Moyenne	Résultat maximum
1	1120	17	18
2	1310	11	19
3	1320	10	18

```
SELECT
  section_id,
  AVG(year_result) as "Moyenne",
  MAX(year_result) as "Résultat maximum"
FROM student
GROUP BY section_id
HAVING AVG(year_result) > 8;
```

Partie V : CUBE et ROLLUP

Exercice 2.5.1 – L'utilisation de « ROLLUP » crée des groupes de données en se déplaçant dans une seule direction, partant de la gauche vers la droite par rapport aux colonnes sélectionnées ? (Vrai/Faux)

Exercice 2.5.2 – Le résultat produit par un « ROLLUP » présente les résultats du plus agrégé au moins agrégé ? (Vrai/Faux)

Exercice 2.5.3 – L'opérateur « CUBE » permet de produire moins de sous-totaux qu'avec l'opérateur « ROLLUP » ? (Vrai/Faux)

Exercice 2.5.4 – Avec l'opérateur « CUBE », le nombre de groupes dans le résultat est indépendant du nombre de colonnes sélectionnées dans le « GROUP BY » ? (Vrai/Faux)

Exercice 2.5.5 – L'opérateur « CUBE » ne peut pas être appliqué à la fonction d'agrégation « SUM » ? (Vrai/Faux)

Exercice 2.5.6 – Donner la moyenne exacte des résultats obtenus par les étudiants par section et par cours, ainsi que la moyenne par section uniquement et enfin, la moyenne générale. La liste ainsi produite reprend l'id de section, de cours le résultat moyen (dans une colonne appelée « Moyenne »). Se baser uniquement sur les sections 1010 et 1320

	section_id	course_id	Moyenne
1	1010	EG1020	2
2	1010	EG2210	4,666666666666667
3	1010	NULL	4
4	1320	EG2120	2
5	1320	EG2210	14
6	1320	NULL	10
7	NULL	NULL	6,57142857142857

```
SELECT
  section_id,
  course_id,
  AVG(year_result) as "Moyenne"
FROM student
WHERE section_id IN (1010, 1320)
GROUP BY section_id, course_id WITH ROLLUP;
```

Exercice 2.5.7 – Donner la moyenne exacte des résultats obtenus par les étudiants par cours et par section, ainsi que la moyenne par cours uniquement, puis par section uniquement et enfin, la moyenne générale. La liste ainsi produite reprend l'id de section, de cours le résultat moyen (dans une colonne appelée « Moyenne »). Se baser uniquement sur les sections 1010 et 1320

```
SELECT
  course_id,
  section_id,
  AVG(year_result) as "Moyenne"
FROM student
WHERE section_id IN (1010, 1320)
GROUP BY CUBE (section_id, course_id);
```

```
(SELECT
  course_id,
  section_id,
  AVG(year_result) as "Moyenne"
FROM student
WHERE section_id IN (1010, 1320)
GROUP BY section_id, course_id WITH ROLLUP)
UNION ALL
(SELECT
  course_id,
  section_id,
  AVG(year_result) as "Moyenne"
FROM student
WHERE section_id IN (1010, 1320)
GROUP BY course_id);
```

	course_id	section_id	Moyenne
1	EG1020	1010	2
2	EG2210	1010	4,666666666666667
3	NULL	1010	4
4	EG2120	1320	2
5	EG2210	1320	14
6	NULL	1320	10
7	NULL	NULL	6,57142857142857
8	EG1020	NULL	2
9	EG2120	NULL	2
10	EG2210	NULL	8,4

Exercice 2.5.8 – Ceci clôture la troisième partie DRL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)

Partie VI : Jointures

Exercice 2.6.1 – Donner pour chaque cours le nom du professeur responsable ainsi que la section dont le professeur fait partie

```
SELECT
  c.course_name,
  s.section_name,
  p.professor_name
FROM course c
JOIN professor p
ON c.professor_id = p.professor_id
JOIN section s
ON p.section_id = s.section_id;
```

	course_name	section_name	professor_name
1	Marketing engineering	MSc Management	zidda
2	Marketing management	MSc Economics	decrop
3	Financial Management	BA Sociology	giot
4	Derivatives	BA Sociology	giot
5	Supply chain manage...	MSc Management	scheppens

Exercice 2.6.2 – Donner pour chaque section, l'id, le nom et le nom de son délégué. Classer les sections dans l'ordre inverse des id de section. Un délégué est un étudiant de la table « STUDENT »

```
SELECT
  s.section_id,
  s.section_name,
  st.last_name
FROM section s
JOIN student st
ON s.delegate_id = st.student_id
ORDER BY s.section_id DESC;
```

	section_id	section_name	last_name
1	1320	MA Sociology	Basinger
2	1310	BA Sociology	Reeves
3	1120	MSc Economics	Basinger
4	1110	BSc Economics	Marceau
5	1020	MSc Management	Portman
6	1010	BSc Management	Willis

Exercice 2.6.3 – Donner pour chaque section, le nom des professeurs qui en sont membre

```
SELECT
  s.section_id,
  s.section_name,
  p.professor_name
FROM section s
LEFT JOIN professor p
ON s.section_id = p.section_id
ORDER BY s.section_id DESC;
```

	section_id	section_name	professor_name
1	1320	MA Sociology	NULL
2	1310	BA Sociology	giot
3	1310	BA Sociology	lecourt
4	1120	MSc Economics	decrop
5	1110	BSc Economics	louveaux
6	1020	MSc Management	zidda
7	1020	MSc Management	scheppens
8	1010	BSc Management	NULL

Exercice 2.6.4 – Même objectif que la question 3 mais seuls les sections comportant au moins un professeur doivent être reprises

	section_id	section_name	professor_name
1	1310	BA Sociology	giot
2	1310	BA Sociology	lecourt
3	1120	MSc Economics	decrop
4	1110	BSc Economics	louveaux
5	1020	MSc Management	zidda
6	1020	MSc Management	scheppens

```
SELECT
  s.section_id,
  s.section_name,
  p.professor_name
FROM section s
RIGHT JOIN professor p
ON s.section_id = p.section_id
ORDER BY s.section_id DESC;
```

Exercice 2.6.5 – Donner à chaque étudiant ayant obtenu un résultat annuel supérieur ou égal à 12 son grade en fonction de son résultat annuel et sur base de la table grade. La liste doit être classée dans l'ordre alphabétique des grades attribués

	last_name	year_result	Grade
1	Basinger	19	E
2	Garcia	19	E
3	Gamer	18	E
4	Berry	18	E
5	Connery	12	S
6	Bacon	16	TB
7	Roberts	17	TB

```
SELECT
  s.last_name,
  s.year_result,
  g.grade
FROM student s
JOIN grade g
ON s.year_result BETWEEN g.lower_bound AND g.upper_bound
WHERE year_result >= 12
ORDER BY g.grade ASC;
```

Exercice 2.6.6 – Donner la liste des professeurs et la section à laquelle ils se rapportent ainsi que le(s) cour(s) (nom du cours et crédits) dont le professeur est responsable. La liste est triée par ordre décroissant des crédits attribués à un cours

	professor_name	section_name	course_name	course_ects
1	zidda	MSc Management	Marketing engineering	4.0
2	giot	BA Sociology	Financial Management	4.0
3	decrop	MSc Economics	Marketing management	3.5
4	giot	BA Sociology	Derivatives	3.0
5	scheppens	MSc Management	Supply chain management et e-business	2.5
6	louveaux	BSc Economics	NULL	NULL
7	lecourt	BA Sociology	NULL	NULL

```
SELECT
  p.professor_name,
  s.section_name,
  c.course_name,
  c.course_ects
FROM professor p
JOIN section s
ON p.section_id = s.section_id
LEFT JOIN course c
ON p.professor_id = c.professor_id
ORDER BY c.course_ects DESC;
```

Exercice 2.6.7 – Donner pour chaque professeur son id et le total des crédits ECTS (« ECTS_TOT ») qui lui sont attribués. La liste proposée est triée par ordre décroissant de la somme des crédits alloués

```
SELECT
  p.professor_id,
  SUM(c.course_ects) AS ECTS_TOT
FROM professor p
LEFT JOIN course c
ON p.professor_id = c.professor_id
GROUP BY p.professor_id
ORDER BY ECTS_TOT DESC;
```

	professor_id	ECTS_TOT
1	3	7.0
2	1	4.0
3	2	3.5
4	5	2.5
5	6	NULL
6	4	NULL

Exercice 2.6.8 – Donner la liste (nom et prénom) de l'ensemble des professeurs et des étudiants dont le nom est composé de plus de 8 lettres. Ajouter une colonne pour préciser la catégorie (S pour « STUDENT », P pour « PROFESSOR ») à laquelle appartient l'individu

```
SELECT
  professor_surname AS first_name,
  professor_name AS last_name,
  "P" AS "Catégorie"
FROM professor
WHERE CHAR_LENGTH(professor_name) > 8
UNION
SELECT
  first_name,
  last_name,
  "S"
FROM student
WHERE CHAR_LENGTH(last_name) > 8;
```

	first_name	last_name	Catégorie
1	georges	scheppens	P
2	Reese	Witherspoon	S
3	Sarah	Michelle Gellar	S

Exercice 2.6.9 – Donner l'id de chacune des sections qui n'ont pas de professeur attribué

	section_id
1	1010
2	1320

```
SELECT
  s.section_id
FROM section s
LEFT JOIN professor p ON s.section_id = p.section_id
WHERE p.professor_id IS NULL;
```

Exercice 2.6.10 – Ceci clôture la quatrième partie DRL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)

Partie VII : Requêtes imbriquées

Exercice 2.7.1 – Donner la liste des étudiants (nom et prénom) qui font partie de la même section que mademoiselle « Roberts ». La liste doit être classée par ordre alphabétique sur le nom et mademoiselle « Roberts » ne doit pas apparaître dans la liste

```
SELECT
  last_name,
  first_name,
  section_id
FROM student
WHERE section_id = (SELECT
  section_id
  FROM student
  WHERE last_name LIKE "roberts")
AND last_name NOT LIKE "roberts";
```

	last_name	first_name	section_id
1	Bacon	Kevin	1120
2	Gamer	Jennifer	1120

Exercice 2.7.2 – Donner la liste des étudiants (nom, prénom et résultat) de l'ensemble des étudiants ayant obtenu un résultat annuel supérieur au double du résultat moyen pour l'ensemble des étudiants

```
SELECT
  s.last_name,
  s.first_name,
  s.year_result
FROM student s
WHERE s.year_result > 2*(SELECT AVG(year_result) FROM student);
```

	last_name	first_name	year_result
1	Basinger	Kim	19
2	Roberts	Julia	17
3	Garcia	Andy	19
4	Gamer	Jennifer	18
5	Berry	Halle	18

Exercice 2.7.3 – Donner la liste de toutes les sections qui n'ont pas de professeur

```
SELECT
  section_id,
  section_name
FROM section
WHERE section_id NOT IN (
  SELECT
    section_id
  FROM professor
  GROUP BY section_id
);
```

	section_id	section_name
1	1010	BSc Management
2	1320	MA Sociology

Exercice 2.7.4 – Donner la liste des étudiants qui ont comme mois de naissance le mois correspondant à la date d’engagement du professeur « Giot ». Classer les étudiants par ordre de résultat annuel décroissant

```
SELECT
  s.last_name,
  s.first_name,
  DATE_FORMAT(s.birth_date, "%m/%d/%Y") as "Date de Naissance",
  s.year_result
FROM student s
WHERE MONTH(s.birth_date) = (
  SELECT
    MONTH(p.professor_hire_date)
  FROM professor p
  WHERE p.professor_name IN ("giot")
);
```

	last_name	first_name	Date de Naissance	year_result
1	Basinger	Kim	12/08/1953	19
2	Milano	Alyssa	12/19/1972	7

Exercice 2.7.5 – Donner la liste des étudiants qui ont obtenu le grade « TB » pour leur résultat annuel

```
SELECT
  last_name,
  first_name,
  year_result
FROM student
WHERE student_id IN (
  SELECT
    s.student_id
  FROM student s
  JOIN grade g ON s.year_result BETWEEN g.lower_bound AND g.upper_bound
  WHERE g.grade IN ("TB")
);
```

	last_name	first_name	year_result
1	Bacon	Kevin	16
2	Roberts	Julia	17

Exercice 2.7.6 – Donner la liste des étudiants qui appartiennent à la section pour laquelle Mademoiselle « Marceau » est déléguée

```
SELECT
  s.last_name,
  s.first_name,
  s.section_id
FROM student s
WHERE s.section_id IN (
  SELECT
    sec.section_id
  FROM section sec
  JOIN student stu ON sec.delegate_id = stu.student_id
  WHERE stu.last_name IN ("Marceau")
);
```

	last_name	first_name	section_id
1	De Niro	Robert	1110
2	Depp	Johnny	1110
3	Garcia	Andy	1110
4	Marceau	Sophie	1110
5	Milano	Alyssa	1110
6	Morse	David	1110

Exercice 2.7.7 – Donner la liste des sections qui se composent de plus de quatre étudiants

```
SELECT
  sec.section_id,
  sec.section_name
FROM section sec
WHERE sec.section_id IN (
  SELECT section_id FROM student
  GROUP BY section_id
  HAVING COUNT(section_id) > 4
);
```

	section_id	section_name
1	1020	MSc Management
2	1110	BSc Economics

Exercice 2.7.8 – Donner la liste des étudiants premiers de leur section en terme de résultat annuel et qui n'appartiennent pas aux sections dont le résultat moyen est inférieure à 10

```
SELECT s.student_id, s.last_name, s.section_id FROM student s
WHERE s.year_result = (
  SELECT
    MAX(s2.year_result)
  FROM student s2
  WHERE s2.section_id = s.section_id
) AND s.section_id NOT IN (
  SELECT
    section_id
  FROM student
  GROUP BY section_id
  HAVING AVG(year_result) < 10
)
```

	last_name	first_name	section_id
1	Berry	Halle	1320
2	Basinger	Kim	1310
3	Gamer	Jennifer	1120

Exercice 2.7.9 – Donner la section qui possède la moyenne la plus élevée. Le résultat présente le numéro de section ainsi que sa moyenne

	section_id	AVG
1	1120	17

Exercice 2.7.10 – Ceci clôture la cinquième partie DRL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)

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

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)

Exercice 3.3 – Créer une table « section_archives » qui contiendra une copie des données contenues dans la table 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. Les valeurs de l'id de section et du cours devront être récupérées là où elles se trouvent dans la base de données, sans les renseigner directement

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 est 1010 (vous ne connaissez pas la valeur de l'ID de ce délégué)

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

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)

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

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

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)

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

Exercice 3.12 – Supprimer votre voisin de la base de données

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 ?

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

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

Exercice 3.16 (bonus DDL-DML) – 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

Exercice 3.17 – Ceci clôture la partie DDL du cours. Avant de passer à la suite de la matière, nous vous invitons à prendre un peu de temps afin d'évaluer personnellement votre niveau de compréhension de la matière en vous référant aux derniers slides du module (slides d'auto-évaluation)