



Version expérimentale  
En cours de validation



## TRAVAUX PRATIQUES – FILIÈRE DÉVELOPPEMENT DIGITAL Option Web Full Stack

### M203 – Gérer les données

Equipe de rédaction:  
Mme Widad Jakjoud



45 heures



## 1. Exploiter les fonctionnalités

### avancées d'un SGBD relationnel

Maitriser le langage de programmation  
procédurale sous MySQL  
Optimiser une base de données MySQL  
Protéger la base de données MySQL

## 2. Exploiter les fonctionnalités des

### bases de données NoSQL MongoDB

Découvrir les bases de données NoSQL  
Mettre en place une base de données MongoDB  
Modéliser les documents  
Manipuler les données avec mongoDB  
Effectuer des requêtes depuis des programmes  
Python  
Sécuriser une base de données MongoDB



# PARTIE 1

## Exploiter les fonctionnalités avancées d'un SGBD relationnel

### Dans cette partie, vous allez :

- Maîtriser le langage de programmation procédurale sous MySQL
- Optimiser une base de données MySQL
- Pouvoir protéger la base de données MySQL



25 heures



# Activité 1

## Rappel du langage SQL

### Compétences visées :

- Installer l'éditeur Workbench
- Rappeler le langage de définition des données (LDD),
- Rappeler le langage de manipulation des données (LMD),

### Recommandations clés :

- Suivre les instructions du TP et organiser le dossier de travail
- Utiliser le résumé théorique pour réaliser le projet de synthèse



**05 heures**

# CONSIGNES

## 1. Pour le formateur :

- Demander aux apprenants de suivre les étapes décrites dans le résumé théorique du cours et d'appliquer les procédures
- Demander aux apprenants de réaliser le travail de synthèse

## 2. Pour l'apprenant :

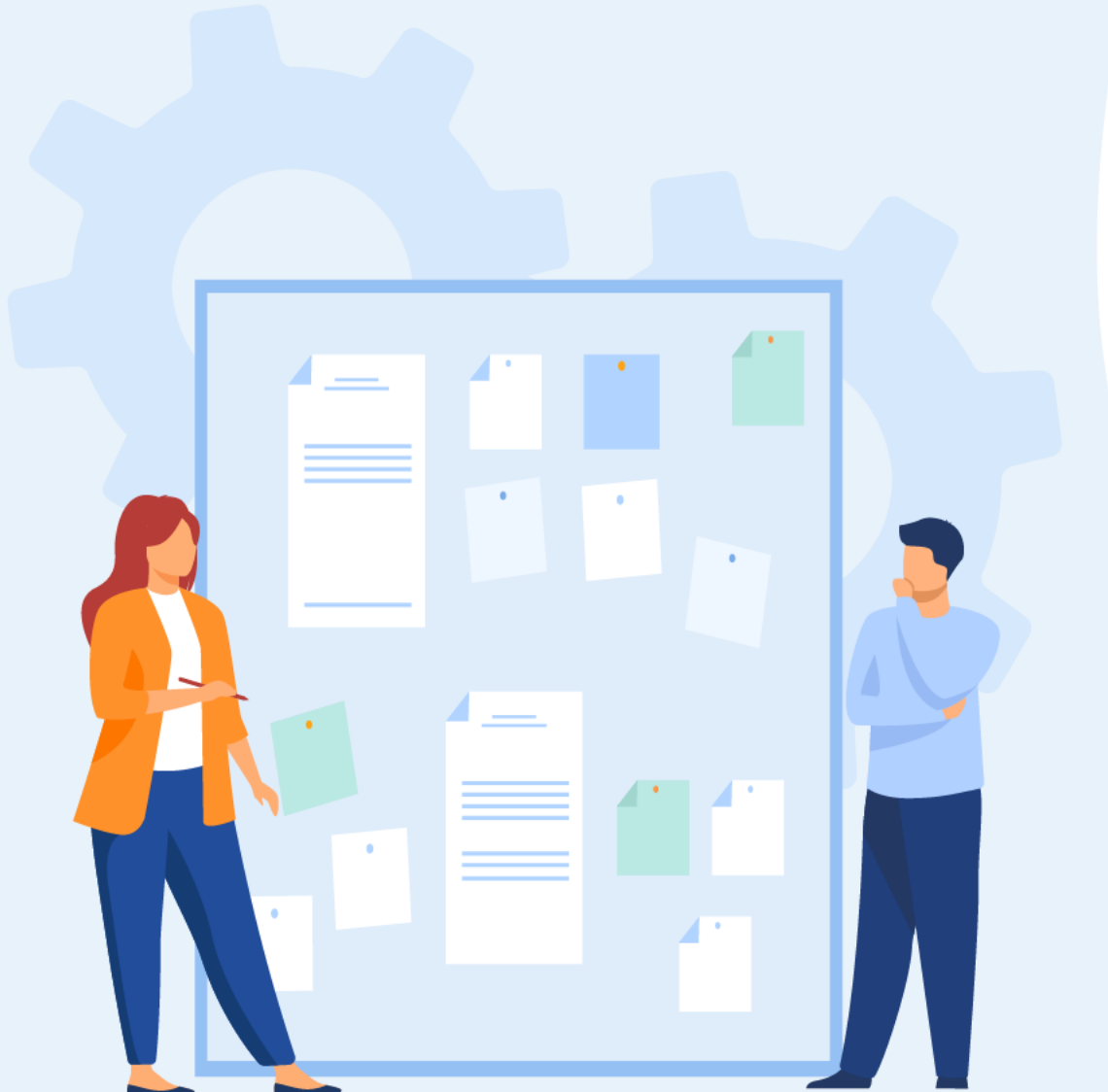
- Installer Workbench en suivant les instructions du formateur
- Créer un dossier de travail dans Workbench et y créer les fichiers de réalisation

## 3. Conditions de réalisation :

- Support de résumé théorique accompagnant

## 4. Critères de réussite :

- Le stagiaire est-il capable de :
  - Créer une base de données,
  - Créer les objets d'une base de données
  - Ecrire des requêtes de manipulation des données (INSERT, UPDATE, DELETE et SELECT,..)



# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse

Soit le MLD suivant:

- **Developpeur** (NumDev, NomDev, AdrDev, EmailDev, TelDev)
- **Projet** (NumProj, TitreProj, DateDeb, DateFin)
- **Logiciel** (CodLog, NomLog, PrixLog, #NumProj)
- **Realisation** (#NumProj, #NumDev)

#### 1. Partie description des données

- Créer la base de données **Glogiciels** en prenant en considération que les champs précédés de # **sont des clés étrangères** et les champs soulignés **sont des clés primaires**.

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher les noms et les prix des logiciels appartenant au projet ayant comme titre « gestion de stock », triés dans l'ordre décroissant des prix
- Afficher le titre du projet et le total des prix des logiciels du projet numéro 10.
- Afficher le nombre de développeurs par projet
- Afficher les projets qui ont plus que 5 logiciels
- Afficher les numéros et noms des développeurs qui ont participé dans le développement de tous les projets.
- Afficher les numéros de projets dans lesquelles tous les développeurs y participent dans sa réalisation.

**Developpeur** (NumDev, NomDev, AdrDev, EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 1. Partie description des données

- La base de données **Glogiciels**

```
create DATABASE DevLogiciels;
```

```
use DevLogiciels;
```

```
create table developpeur(numdev int primary key, nomdev varchar(20), adrDev varchar(20), emailDev  
varchar(20), telDev int);
```

```
create table projet(numproj int primary key, titreProj varchar(20),dateDeb datetime, dateFin datetime);
```

```
create table logiciel (codeLog int primary key, nomLog varchar(20), prixLog decimal, numProj int, foreign key  
(numProj) references projet(numProj));
```

```
create table realisation (numProj int , numDev int, foreign key (numProj) references projet(numproj), foreign key  
(numdev) references developpeur(numdev));
```

**Developpeur** (NumDev, NomDev, AdrDev,  
EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)



# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher les noms et les prix des logiciels appartenant au projet ayant comme titre « gestion de stock », triés dans l'ordre décroissant des prix

```
select nomlog, prixlog
```

```
from logiciel, projet
```

```
where logiciel.numProj = projet.numProj and titreProj='gestion du stock'
```

```
order by prixLog desc;
```

**Developpeur** (NumDev, NomDev, AdrDev, EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher les noms et les prix des logiciels appartenant au projet ayant comme titre « gestion de stock », triés dans l'ordre décroissant des prix

```
select nomlog, prixlog  
  
from logiciel  
  
where numProj =  
  
    (select numproj  
  
        from projet  
  
        where titreProj='gestion du stock')  
  
order by prixLog desc;
```

**Developpeur** (NumDev, NomDev, AdrDev, EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher le titre du projet et le total des prix des logiciels du projet numéro 10.

```
select titreProj, sum(prixlog) as 'somme des prix'  
from logiciel, projet  
where logiciel.numProj = projet.numProj  
and logiciel.numProj = 1;
```

**Developpeur** (NumDev, NomDev, AdrDev,  
EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher le nombre de développeurs par projet

```
select count(*) as'Nombre de developpeur',titreProj  
from projet, realisation  
where projet.numProj = realisation.numProj  
group by titreProj;
```

**Developpeur** (NumDev, NomDev, AdrDev,  
EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher les projets qui ont plus que 5 logiciels

```
select numProj, count(*) as 'nombres de logiciels'  
  
from logiciel  
  
group by numProj  
  
having count(*)>=5;
```

**Developpeur** (NumDev, NomDev, AdrDev,  
EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher les numéros et noms des développeurs qui ont participé dans le développement de tous les projets.

```
select developpeur.numDev, nomDev  
from developpeur, realisation  
where developpeur.numDev = realisation.numdev  
group by developpeur.numDev,nomDev  
having count(*) = (select count(*) from projet);
```

**Developpeur** (NumDev, NomDev, AdrDev,  
EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)

# Activité 1

## Réalisation des requêtes simples et fonctions d'agrégation



### Travail de synthèse (Correction)

#### 2. Partie manipulation des données

Exprimer les requêtes suivantes dans le langage SQL

- Afficher les numéros de projets dans lesquelles tous les développeurs y participent dans sa réalisation

```
select numProj  
  
from realisation  
  
group by numProj  
  
having count(*) = (select count(numDev)  
  
from developpeur);
```

**Developpeur** (NumDev, NomDev, AdrDev,  
EmailDev, TelDev)

**Projet** (NumProj, TitreProj, DateDeb, DateFin)

**Logiciel** (CodLog, NomLog, PrixLog, #NumProj)

**Realisation** (#NumProj, #NumDev)



## Activité 2

### Créer les procédures stockées et les fonctions

#### Compétences visées :

- Manipuler les instructions de base du langage procédural de MySql
- Manipuler les types de programmes sous le langage procédural MySQL,

#### Recommandations clés :

- Suivre les instructions du TP et organiser le dossier de travail
- Utiliser le résumé théorique pour réaliser le projet de synthèse



**10 heures**



# CONSIGNES

## 1. Pour le formateur :

- Demander aux apprenants de suivre les étapes décrites dans le résumé théorique du cours et d'appliquer les procédures
- Demander aux apprenants de réaliser le travail de synthèse

## 2. Pour l'apprenant :

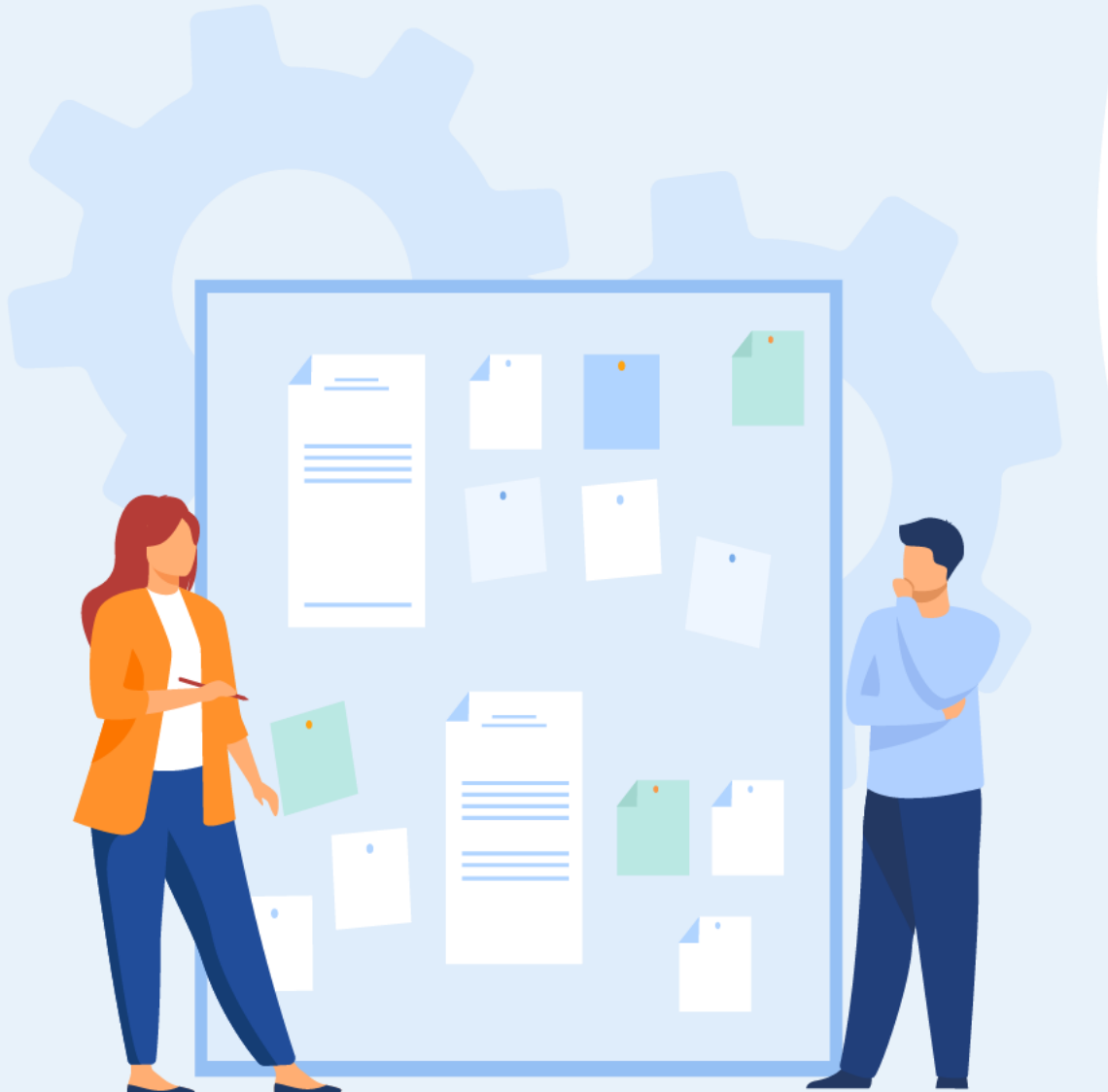
- Installer Workbench en suivant les instructions du formateur
- Créer un dossier de travail dans Workbench et y créer les fichiers de réalisation

## 3. Conditions de réalisation :

- Support de résumé théorique accompagnant

## 4. Critères de réussite :

- Le stagiaire est-il capable de :
  - Créer une procédure stockées en distinguant son type,
  - Ecrire un script en manipulant les structures de contrôles du langage,
  - Créer des fonctions et les utiliser



## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 1. Procédure stockée sans paramètres

- Créer une procédure stockée nommée **sp\_ValeursProduits** qui affiche la liste des produits avec pour chaque produit sa référence, son intitulé et sa valeur en stock (**quantite** et **stock \* prix unitaire**)

```
Client(numCl, nomCl, prCl, telCl)
Commande(numBon, dateCmd, #numCl)
Produit (reference, intitule, PU, quantiteStock)
LigneCommande(#numBon, #reference,
quantiteCommandee)
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 1. Procédure stockée sans paramètres

- Créer une procédure stockée nommée **sp\_ValeursProduits** qui affiche la liste des produits avec pour chaque produit sa référence, son intitulé et sa valeur en stock (**quantite** et **stock \* prix unitaire**)

```
Client(numCl, nomCl, prCl, telCl)
Commande(numBon, dateCmd, #numCl)
Produit (reference, intitule, PU, quantiteStock)
LigneCommande(#numBon, #reference,
quantiteCommandee)
```

```
delimiter //
```

```
create procedure sp_valeursProduits()
```

```
begin
```

```
select reference, intitule, quantiteStock * pu as 'valeur en stock' from produit ;
```

```
end; //
```

```
call sp_valeursProduits()
```

#### **Delimiter ? :**

Un caractère utilisé par MySQL pour séparer les requêtes et les programmes.

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 2. Procédure stockée avec paramètres d'entrée

- Créer une procédure stockée nommée **sp\_listeProduitsCommandes** qui affiche la liste des produits d'une commande dont le numéro est donné en paramètre :

```
Client(numCl, nomCl, prCl, telCl)
Commande(numBon, dateCmd, #numCl)
Produit (reference, intitule, PU, quantiteStock)
LigneCommande(#numBon, #reference,
quantiteCommandee)
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 2. Procédure stockée avec paramètres d'entrée

```
Client(numCl, nomCl, prCl, telCl)
Commande(numBon, dateCmd, #numCl)
Produit (reference, intitule, PU, quantiteStock)
LigneCommande(#numBon, #reference,
quantiteCommandee)
```

- Créer une procédure stockée nommée **sp\_listeProduitsCommandes** qui affiche la liste des produits d'une commande dont le numéro est donné en paramètre :

```
delimiter //
```

```
create procedure sp_listeProduitsCommandes(IN num int)
```

```
Begin
```

```
    select * from produit where reference in (select reference from lignecommande where numbon=num);
```

```
end; //
```

```
call sp_listeProduitsCommandes(10)
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 3. Procédure stockée avec paramètre de sortie

- Créer une procédure stockée nommée **sp\_nbCommandes** qui retourne le nombre de commandes de la table commande

```
Client(numCl, nomCl, prCl, telCl)
Commande(numBon, dateCmd, #numCl)
Produit (reference, intitule, PU, quantiteStock)
LigneCommande(#numBon, #reference,
quantiteCommandee)
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 3. Procédure stockée avec paramètre de sortie

- Créer une procédure stockée nommée **sp\_nbCommandes** qui retourne le nombre de commandes de la table commande

```
delimiter //
```

```
create procedure sp_nbCommandes (OUT nb int)
```

```
begin
```

```
select count(*) into nb from commande;
```

```
end//
```

```
Client(numCl, nomCl, prCl, telCl)  
Commande(numBon, dateCmd, #numCl)  
Produit (reference, intitule, PU, quantiteStock)  
LigneCommande(#numBon, #reference,  
quantiteCommandee)
```

#### Execution:

```
call sp_nbCommandes(@nombre);  
select @nombre
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

#### 4. Procédure stockée avec paramètres d'entrée / sortie

```
Client(numCl, nomCl, prCl, telCl)
Commande(numBon, dateCmd, #numCl)
Produit (reference, intitule, PU, quantiteStock)
LigneCommande(#numBon, #reference,
quantiteCommandee)
```

- Créer une procédure stockée nommée **gainActualise** qui prend en paramètre d'entrée le gain mensuel, en paramètre de sortie le chiffre d'affaires puis l'actualise (**chiffre d'affaires + gain**) et le retourne:



## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

#### 4. Procédure stockée avec paramètres d'entrée / sortie

- Créer une procédure stockée nommée **gainActualise** qui prend en paramètre d'entrée le gain mensuel, en paramètre de sortie le chiffre d'affaires puis l'actualise (**chiffre d'affaires + gain**) et le retourne:

```
delimiter //
```

```
create procedure gainActualise (inout ChiffreAffaires float,in gain float)
```

```
begin
```

```
set chiffreAffaires = chiffreAffaires + gain;
```

```
end //
```

```
Client(numCl, nomCl, prCl, telCl)  
Commande(numBon, dateCmd, #numCl)  
Produit (reference, intitule, PU, quantiteStock)  
LigneCommande(#numBon, #reference,  
quantiteCommandee)
```

#### Execution:

```
set @cc = 150000.00;call  
gainActualise(@cc,2000.00);  
select @cc;
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 5. Suppression d'une procédure stockée

- DROP procedure **gainActualise**

```
40 • call gainActualise(@cc,2000.00);
```

```
41 • select @cc;
```

```
42
```

```
✘ 17 08:38:17 call gainActualise(@cc,2000.00) Error Code: 1305. PROCEDURE.gainActualise does not exist
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Procédures stockées

##### 6. Modification d'une procédure stockée

Il faut la supprimer et la recréer **DROP PROCEDURE** et **CREATE PROCEDURE**.

## Activité 2

### Créer les procédures stockées et les fonctions



#### Fonction

##### 1. Fonction sans paramètres

- Créer une fonction nommée **Bonjour()** qui affiche '**Bonjour de MySQL**'

## Activité 2

### Créer les procédures stockées et les fonctions



#### Fonction

##### 1. Fonction sans paramètres

- Créer une fonction nommée Bonjour() qui affiche 'Bonjour de MySQL'

**delimiter //**

**CREATE FUNCTION Bonjour()**

**RETURNS VARCHAR(20)**

**RETURN 'Bonjour de MySQL';**

**//**

**#Appel:**

**select Bonjour()**

## Activité 2

### Créer les procédures stockées et les fonctions



#### Fonction

##### 2. Fonction avec paramètres

- Ecrire une Fonction **FACTORIELLE** qui prend en paramètre un nombre X puis calcule et retourne sa factorielle:

## Activité 2

### Créer les procédures stockées et les fonctions



## Fonction

### 2. Fonction avec paramètres

- Ecrire une Fonction FACTORIELLE qui prend en paramètre un nombre X puis calcule et retourne sa factorielle:

```
Delimiter //  
CREATE FUNCTION Factorielle (x int)  
RETURNS float  
BEGIN  
    DECLARE i INT DEFAULT 1 ;  
    DECLARE f float DEFAULT 1;  
    WHILE i <= x DO  
        SET f = f * i;  
        SET i = i + 1;  
    END WHILE ;  
    RETURN f;  
END ;//
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Fonction

##### 2. Fonction avec paramètres

- Ecrire une fonction *Disponibilite* qui prend en paramètre la référence d'un produit puis retourne « Produit disponible » si la quantité en stock est supérieure à 10, et « Besoin en réapprovisionnement » sinon

```
delimiter //
CREATE FUNCTION disponibilite(r int)
RETURNS varchar(20)
BEGIN
    declare qs int;
    select quantiteStock into qs from produit where reference = r;
    if (qs<20) then
        RETURN 'Rupture en Stock';
    else RETURN 'Produit dispo';
    end if;
END//
```



## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse

"*CaftanModerne*" est une organisation qui organise un défilé de mode international. Dans ce défilé, des costumes défilent devant un jury professionnel composé de plusieurs membres. Chaque membre va attribuer une note à chaque costume.

On propose la structure suivante de la base de données:

**Styliste**(NumStyliste, NomStyliste, AdrStyliste)

**Caftan**(numCaftan, DesignationCaftan, # NumStyliste)

**MembreJury** (numMembreJury, nom, fonction)

**NotesJury**(#NumCaftan, #NumMembre, note)

**Fonction**(Fonction)

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse

Il est demandé de créer les procédures suivantes :

1. Une procédure Qui affiche la liste des caftans avec pour chaque caftan le numéro, la désignation, le nom et l'adresse du styliste qui l'a réalisé
2. Une procédure qui reçoit un numéro de caftan et qui affiche la désignation, le nom et l'adresse du styliste concerné
3. Une procédure qui reçoit un numéro de caftan et qui affiche la liste des notes attribuées avec pour chaque note le numéro du membre de jury qui l'a attribué, son nom, sa fonction et la note.
4. Une fonction qui retourne le nombre total de caftans
5. Une procédure qui reçoit un numéro de caftan et un numéro de membre de jury et qui retourne la note que ce membre a attribuée à ce caftan
6. Une fonction qui reçoit un numéro de caftan et qui retourne sa moyenne.

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

```
create table Styliste( NumStyliste int primary key, NomStyliste varchar(10), AdrStyliste varchar(20));
```

```
create table Caftan(NumCaftan int primary key, designationCaftan varchar(20), numstyliste int, foreign key (numstyliste) references styliste(numstyliste));
```

```
create table MembreJury (numMembreJury int primary key, nom varchar(20), fonction varchar(20), foreign key (fonction) references MembreJury(fonction));
```

```
create table NotesJury(numCaftan int, NumMembre int, note double, foreign key (NumMembre) references MembreJury(numMembreJury), primary key (numCaftan, NumMembre));
```

```
create table Fonction (fonction varchar(20))
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

```
create table Styliste( NumStyliste int primary key, NomStyliste varchar(10), AdrStyliste varchar(20));
```

```
create table Caftan(NumCaftan int primary key, designationCaftan varchar(20), numstyliste int, foreign key (numstyliste) references  
styliste(numstyliste));
```

```
create table MembreJury (numMembreJury int primary key, nom varchar(20), fonction varchar(20), foreign key (fonction) references  
MembreJury(fonction));
```

```
create table NotesJury(numCaftan int, NumMembre int, note double, foreign key (NumMembre) references  
MembreJury(numMembreJury), primary key (numCaftan, NumMembre));
```

```
create table Fonction (fonction varchar(20))
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

1. Une procédure qui affiche la liste des caftans avec pour chaque caftan le numéro, la désignation, le nom et l'adresse du styliste qui l'a réalisé

```
Delimiter //
```

```
create procedure Q1 ()
```

```
Begin
```

```
    select NumCaftan, designationCaftan, nomStyliste, adrStyliste
```

```
    from caftan c, styliste s
```

```
    where c.numStyliste =s.numstyliste;
```

```
end//
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

2. Une procédure Qui reçoit un numéro de costume et qui affiche la désignation, le nom et l'adresse du styliste concerné

```
delimiter //  
create procedure Q2(IN num int)  
Begin  
  
    select designationCaftan, nomStyliste, adrStyliste  
  
    from caftan c, styliste s  
  
    where c.numstyliste = s.numStyliste  
  
    and numcaftan = @num;  
  
end //
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

3. Une fonction qui reçoit un numéro de caftan et numéro de membre de jury et retourne la note que ce membre a attribuée à ce caftan

```
delimiter //  
create function Q3( numC int, numJ int)  
returns int  
    begin  
        declare note float;  
        select note into note from NotesJury where numCaftan=numC and numMembre = numJ;  
    return note;  
end//
```

## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

4. Une fonction qui reçoit un numéro de caftan et retourne sa note moyenne

```
delimiter //  
create function Q4( numC int)  
returns int  
    begin  
        declare moyenne float;  
        select avg(note) into moyenne from NotesJury where numCaftan=numC;  
    return moyenne;  
end//
```



## Activité 2

### Créer les procédures stockées et les fonctions



#### Travail de synthèse (Correction)

5. Une procédure qui affiche pour chaque caftan (sa désignation) la moyenne attribuée par les membres de jury

```
delimiter //
```

```
create procedure Q5()
```

```
Begin
```

```
    select designationCaftan,avg(note) as "Moyenne«
```

```
    from NotesJury,Caftan
```

```
    where caftan.numCaftan = notesjury.numCaftan
```

```
    group by NotesJury.numcaftan;
```

```
end//
```



## Activité 3

### Créer les déclencheurs

#### Compétences visées :

- Manipuler les types de programmes sous le langage procédural MySQL,

#### Recommandations clés :

- Suivre les instructions du TP et organiser le dossier de travail
- Utiliser le résumé théorique pour réaliser le projet de synthèse



**10 heures**

# CONSIGNES

## 1. Pour le formateur :

- Demander aux apprenants de suivre les étapes décrites dans le résumé théorique du cours et d'appliquer les procédures
- Demander aux apprenants de réaliser le travail de synthèse

## 2. Pour l'apprenant :

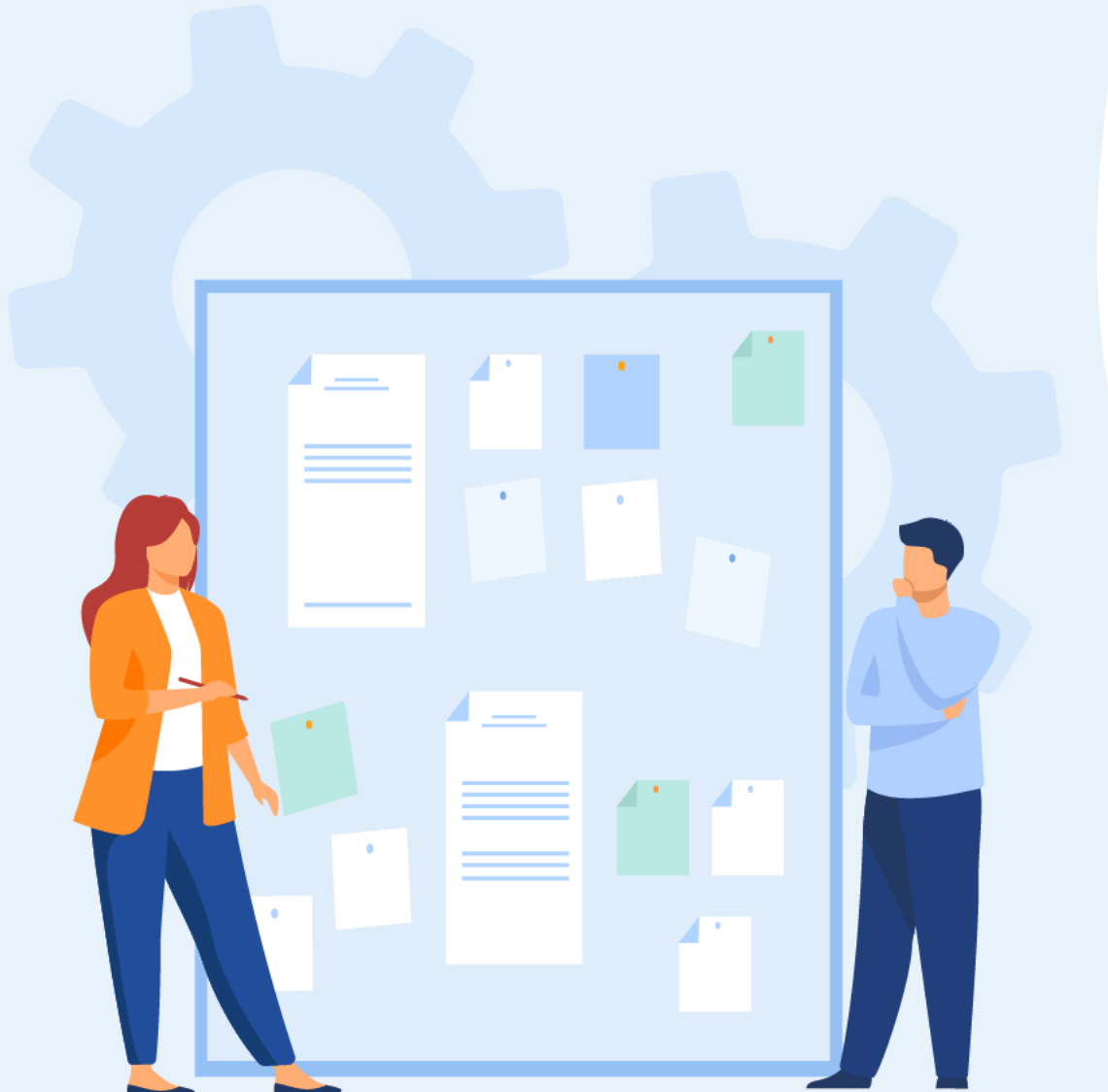
- Installer Workbench en suivant les instructions du formateur
- Créer un dossier de travail dans Workbench et y créer les fichiers de réalisation

## 3. Conditions de réalisation :

- Support de résumé théorique accompagnant

## 4. Critères de réussite :

- Le stagiaire est-il capable de :
  - Créer un déclencheur en distinguant son type,
  - Manipuler les déclencheurs ainsi créés



## Activité 3

### Créer les déclencheurs



#### Travail de synthèse

Soit le MLD de la base de données **GComptesBancaire**:

**Compte**(NumCompte, solde, TypeCompte, #NumCl)

**Client**(CIN, nom, prenom, adr, tel)

**Operation** ( NumOP, TypeOp, MontantOp, #NumCpt, DateOp)

Avec les contraintes suivantes

- Le numéro de l'opération est automatique ,
- La date d'opération prend par défaut la date du jour,
- Un compte ne peut être que de type Compte Courant (CC) ou Compte d'Epargne (CE) : Ajouter une contrainte pour que le champ TypeCompte ne peut prendre que deux valeurs CC ou CE.
- Un client ne peut avoir qu'un seul compte courant mais plusieurs comptes d'épargne

## Activité 3

### Créer les déclencheurs



#### Travail de synthèse

Soit le MLD de la base de données **GComptesBancaire**:

**Compte**(NumCompte, solde, TypeCompte, #NumCl)

**Client**(CIN, nom, prenom, adr, tel)

**Operation** ( NumOP, TypeOp, MontantOp, #NumCpt, DateOp)

Avec les contraintes suivantes

- Le numéro de l'opération est automatique ,
- La date d'opération prend par défaut la date du jour,
- Un compte ne peut être que de type Compte Courant (CC) ou Compte d'Epargne (CE) : Ajouter une contrainte pour que le champ TypeCompte ne peut prendre que deux valeurs CC ou CE.
- Un client ne peut avoir qu'un seul compte courant mais plusieurs comptes d'épargne

## Activité 3

### Créer les déclencheurs



#### Travail de synthèse

1. Créer un déclencheur TR\_AJOUT\_COMPTE qui, à la création d'un nouveau compte de type CC, vérifie si le solde est >1500.00 DH,

```
delimiter //create trigger AJOUT_COMPTE
```

```
before insert
```

```
on compte
```

```
for each row
```

```
if NEW.solde <1500 then
```

```
SIGNAL SQLSTATE '45000'
```

```
SET MESSAGE_TEXT = 'Impossible de creer un compte avec un solde initial inferieur à 1500.00Dh.';
```

```
END IF//
```

## Activité 3

### Créer les déclencheurs



#### Travail de synthèse

2. Créer un déclencheur qui empêche la création d'un nouveau compte de type CC pour un client qui en a déjà un

```
delimiter //  
create trigger AJOUT_COMPTE before insert on compte  
for each row  
Begin  
    DECLARE nbCC int;select count(*) into @nbCC from compte where numcl = NEW.numcl and typeCompte='CC';  
    if @nbCC >1 then  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Impossible de creer plus qu un compte courant pour le même client !!';  
    end if;  
end//
```

## Activité 3

### Créer les déclencheurs



#### Travail de synthèse

3. Créer un déclencheur qui empêche la suppression d'un compte dont le solde n'est pas 0

```
delimiter //  
create trigger SUPPRESSION_COMPTE before DELETE on compte  
for each row  
Begin  
    if OLD.solde>0 then  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Impossible de supprimer un compte dont le solde est >0 !!';  
    end if;  
end//
```



## Activité 3

### Créer les déclencheurs



#### Travail de synthèse

3. Créer un déclencheur TR\_UPDATE\_COMPTE qui interdit la modification du type de compte des comptes auxquels sont associées des opérations,

```
delimiter //  
create trigger UPDATE_COMPTE before UPDATE on compte  
for each row  
begin  
    DECLARE nbOp int;  
    select count(*) into @nbOp from operation where numcpt = OLD.numcmp;  
    if @nbOp >0 then  
        SIGNAL SQLSTATE '45000'  
        SET MESSAGE_TEXT = 'Impossible de supprimer un compte qui a des opérations!!';  
    end if;  
end//
```



## PARTIE 2

# Exploiter les fonctionnalités des bases de données NoSQL MongoDB

**Dans cette partie, vous allez :**

- Mettre en place une base de données MongoDB
- Manipuler les documents JSON



**20 heures**



## Activité 1

### Mettre en place une base de données MongoDB

#### Compétences visées :

- Créer une base de données
- Créer une collection de documents

#### Recommandations clés :

- Révision générale du résumé théorique



10 heures

# CONSIGNES

## 1. Pour le formateur :

- Rappeler les bases théoriques sur la création des formulaires
- Définir les différents composants d'un formulaire

## 2. Pour l'apprenant :

- Mettre en place les différents composants d'un formulaire
- Organiser les formulaires en utilisant les tableaux
- Définir les valeurs des différentes propriétés des composants

## 3. Conditions de réalisation :

- Support de résumé théorique accompagnant

## 4. Critères de réussite :

- Le stagiaire est-il capable de :
  - Définir la structure d'un formulaire
  - Mettre en place les éléments d'un formulaire
  - Déterminer les valeurs des propriétés des composants



# Activité 1

## Créer une base de données



### Base de données DBSportifs

Soit la base de données *DBSportifs*,

Créer une collection *Sportif* avec un ensemble de documents représentant un ensemble de sportifs marocains, par exemple :

```
{_id:"sp1",  
"nom":"Radi",  
"prenom":"Abdessalam",  
"genre":"homme",  
"sport":{  
  "description":"marathon",  
  "olympique":"true"}  
}
```

```
{_id:"sp2",  
"nom":"Larbi",  
"prenom":"Benbarek",  
"genre":"homme",  
"sport":{  
  "description":"football",  
  "olympique":"true"}  
}
```

```
{_id:"sp3",  
"nom":"ELGourch",  
"prenom":"Mohamed",  
"genre":"homme",  
"nbMedailles":2,  
"sport":{  
  "description":"cyclisme",  
  "olympique":"true"}  
}
```

```
{_id:"sp4",  
"nom":"Rabii",  
"prenom":"Mohamed",  
"genre":"homme",  
"nbMedailles":3,  
"sport":{  
  "description":"box",  
  "olympique":"true"}  
}
```

```
{_id:"sp5",  
"nom":"elguerrouj",  
"prenom":"Hicham",  
"genre":"homme",  
"nbMedailles":4,  
"sport":{  
  "description":"athletisme",  
  "olympique":"true"}  
}
```

```
{_id:"sp6",  
"nom":"Abissourour",  
"prenom":"Sara",  
"genre":"femme",  
"sport":{  
  "description":"volley ball",  
  "olympique":"true"}}}
```

## Activité 1

### Créer une base de données



#### Base de données DBSportifs (correction)

```
//Création de la base de données
```

```
use DBSportifs
```

```
//création de la collection
```

```
db.createCollection("Sportif")
```

# Activité 1

## Créer une base de données



### Base de données DBSportifs

//Création des documents

- ```
db.getCollection("Sportif").insert(
  {_id:"sp1",
  "nom":"Radi",
  "prenom":"Abdessalam",
  "genre":"homme",
  "sport":{"description":"marathon","olympique":"true"}
})
```
- ```
db.getCollection("Sportif").insert(
  {_id:"sp2",
  "nom":"Larbi",
  "prenom":"Benmbarek",
  "genre":"homme",
  "sport":{"description":"football","olympique":"true"}
})
```

# Activité 1

## Créer une base de données



### Base de données DBSportifs

//Création des documents

- ```
db.getCollection("Sportif").insert(
  {_id:"sp3",
  "nom":"ELGourch",
  "prenom":"Mohamed",
  "genre":"homme",
  "sport":{"description":"cyclisme","olympique":"true"}
})
```
- ```
db.getCollection("Sportif").insert(
  {_id:"sp4",
  "nom":"Bidouane",
  "prenom":"Nezha",
  "genre":"femme",
  "sport":{"description":"athletisme","olympique":"true"}
})
```



# Activité 1

## Créer une base de données



### Base de données DBSportifs

//Création des documents

- ```
db.getCollection("Sportif").insert(
  {_id:"sp5",
  "nom":"ELGaraa",
  "prenom":"Najat",
  "genre":"femme",
  "sport":{"description":"athletisme","olympique":"true"}
})
```
- ```
db.getCollection("Sportif").insert(
  {_id:"sp6",
  "nom":"Rabii",
  "prenom":"Mohamed",
  "genre":"homme",
  "sport":{"description":"box","olympique":"true"}
})
```

# Activité 1

## Créer une base de données



### Base de données DBSportifs

//Création des documents

- `db.getCollection("Sportif").insert(  
 {_id:"sp7",  
 "nom":"elguerrouj",  
 "prenom":"Hicham",  
 "genre":"homme",  
 "sport":{"description":"box","olympique":"true"}}  
 )`
- `db.getCollection("Sportif").insert(  
 {_id:"sp8",  
 "nom":"Abissourour",  
 "prenom":"Sara",  
 "genre":"femme",  
 "sport":{"description":"volley ball","olympique":"true"}}  
 )`

## Activité 1

### Créer une base de données



#### Base de données DBSportifs

//Création des documents

- ```
db.getCollection("Sportif").insert(
  {_id:"sp9","nom":"Belafrikh",
  "prenom":"Amine",
  "genre":"homme",
  "sport":{"description":"Muay Thai","olympique":"false"}
})
```
- ```
db.getCollection("Sportif").insert(
  {_id:"sp10",
  "nom":"Moutawakil",
  "prenom":"Naoual",
  "genre":"femme",
  "nbMedailles":4,
  "sport":{"description":"athletisme","olympique":"true"}
})
```

## Activité 1

### Requêtes sur les documents



#### Effectuer des recherches sur les données

1. Répondre aux requêtes suivantes :
  - a. Quels sont les sportifs (identifiant , nom et prénom) de la base de données?
  - b. Quels sont les sportifs (identifiant, nom et prénom) de genre « Homme »?
  - c. Quel(s) sont les sportifs (identifiant , nom, prénom et genre) qui pratiquent le cyclisme?
  - d. Quels sports ne sont pas olympiques?
  - e. Quel est le premier sportif qui joue au football?

## Activité 1

### Requêtes sur les documents



#### Effectuer des recherches sur les données (correction)

a) Quels sont les sportifs de la base de données?

a) `db.getCollection("Sportif").find()`

b) Quels sont les sportifs ( nom et prenom) de genre « Homme »?

a) `db.getCollection("Sportif").find(`

b)  `{"genre":"homme"},`

c)  `{"_id":0,"nom":1,"prenom":1}`

d) `)`

## Activité 1

### Requêtes sur les documents



#### Effectuer des recherches sur les données (correction)

a) Quel(s) sont les sportifs (identifiant , nom, prenom et genre) qui pratiquent le cyclisme?

a) `db.getCollection("Sportif").find(`

b)  `{"sport.description":"cyclisme"},`

c)  `{"_id":0,"nom":1,"prenom":1})`

b) Quels sports ne sont pas olympiques?

a) `db.getCollection("Sportif").find(`

b)  `{"sport.olympique":"false"},`

c)  `{"sport.description":1})`

## Activité 1

### Requêtes sur les documents



#### Effectuer des recherches sur les données (correction)

- a) Quel est le premier sportif qui joue au football?

```
db.getCollection("Sportif").findOne(  
  {"sport.description":"football"})
```

## Activité 1

### Requêtes sur les documents



#### Tri et fonctions

1. Afficher les sportifs (nom et prénom) triés par ordre alphabétique des noms,
2. Afficher les description des sports triées par ordre alphabétique décroissant,
3. Afficher le nombre de sportifs de la base de données
4. Afficher les descriptions des différents sport sans dédoublant
5. Afficher les deux premières femmes sportives de la base de données
6. Afficher les sportifs qui ont 3 médailles
7. Afficher les noms et prénoms des sportifs qui ont plus que deux médailles
8. Afficher les noms et prénoms des sportifs qui pratiquent les sports suivants: box, athlétisme et cyclisme
9. Afficher les noms et prénoms des sportifs qui n'ont pas de médailles



## Activité 1

### Requêtes sur les documents



#### Tri et fonctions -Correction

1. Afficher les sportifs (nom et prenom) triés par ordre alphabétique des noms,

```
db.getCollection("Sportif").find().sort({"nom":1})
```

2. Afficher les description des sports triées par ordre alphabétique décroissant,

```
db.getCollection("Sportif").find(  
  {},  
  {"style.description":1}).sort({"style.description":-1})
```

## Activité 1

### Requêtes sur les documents



#### Tri et fonctions -Correction

- Afficher le nombre de sportifs de la base de données

```
db.getCollection("Sportif").find().count
```

- Afficher les descriptions des différents sport sans dédoublant

```
db.getCollection("Sportif").distinct("sport.description")
```

- Afficher les deux premières femmes sportives de la base de données

```
db.getCollection("Sportif").find(  
  {"genre": "femme"},  
  {"nom": 1}).limit(2)
```

## Activité 1

### Requêtes sur les documents



#### Tri et fonctions -Correction

6. Afficher les sportifs qui ont 3 médailles

```
db.getCollection("Sportif").find(  
  {"nbMedailles":{"$eq":3}},  
  {"nom":1,"prenom":1})
```

7. Afficher les noms et prénoms des sportifs qui ont plus que deux médailles

```
db.getCollection("Sportif").find(  
  {"nbMedailles":{"$gte":2}},  
  {"nom":1,"prenom":1})
```

## Activité 1

### Requêtes sur les documents



#### Tri et fonctions -Correction

8. Afficher les noms et prénoms des sportifs qui pratiquent les sports suivants: box, athlétisme et cyclisme

```
db.getCollection("Sportif").find(  
  {"sport.description":{"$in":["athlétisme","box","cyclisme"]}},  
  {"nom":1,"prenom":1})
```

9. Afficher les noms et prénoms des sportifs qui n'ont pas de médailles

```
db.getCollection("Sportif").find(  
  {"nbMedailles":{"$exists:false}},  
  {"nom":1,"prenom":1})
```

## Activité 1

### Requêtes sur les documents



#### Agrégation

1. Retourner les sportifs qui pratiquent le cyclisme
2. Calculer la somme des médailles par sport,
3. Afficher le maximum de médailles par sport

## Activité 1

### Requêtes sur les documents



#### Agrégation (correction)

1. Retourner les sportifs qui pratiquent le cyclisme

```
db.getCollection("Sportif").aggregate(  
  [{$match: {"sport.description" : "cyclisme" }}])
```

2. Calculer la somme des médailles par sport,

```
db.getCollection("Sportif").aggregate(  
  [{$group: {_id: "$sport.description", nombre : {$sum:"$nbMedailles"}}}])
```

3. Afficher le maximum de médailles par sport

```
db.getCollection("Sportif").aggregate(  
  [{$group: {_id: "$sport.description", maximum : {$max:"$nbMedailles"}}}])
```

## Activité 1

### Requêtes sur les documents



#### Mise à jour des documents

1. Mettre à jour le champ **nbMedailles** du sportif « **Rabii Mohamed** » pour y mettre la valeur 2
2. Pour tous les sportifs, ajouter un champ « **nationalité** » avec la valeur « **marocaine** »
3. Supprimer les sportifs qui pratiquent le sport « **Muay Thai** »

## Activité 1

### Requêtes sur les documents



#### Mise à jour des documents (correction)

1. Mettre à jour le champ nbMedailles du sportif « Rabii Mohamed » pour y mettre la valeur 2

```
db.getCollection("Sportif").updateOne(  
  {"nom" : "Rabii", "prenom": "Mohamed"},  
  {$set: {"nbMedailles" : 2}})
```

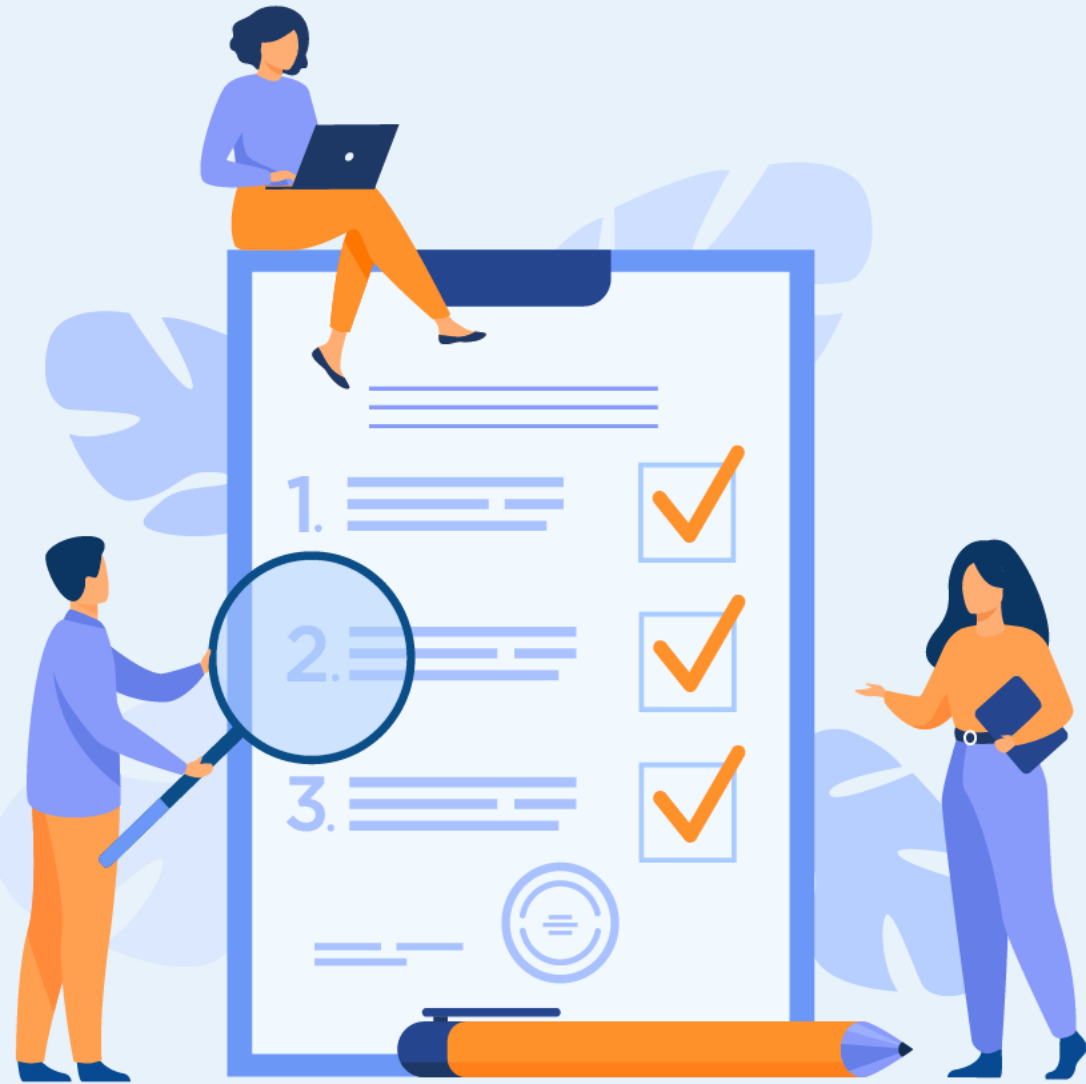
2. Pour tous les sportifs, ajouter un champ « nationalité » avec la valeur « marocaine »

```
db.getCollection("Sportif").updateMany(  
  {}, {$set: {"nationalite": "marocaine"}})
```

3. Supprimer les sportifs qui pratiquent le sport « Muay Thai »

```
db.getCollection("Sportif").remove({"sport.description": "Muay Thai"})
```





## Activité 2

### Effectuer des requêtes depuis des programmes Python

#### Compétences visées :

- Installer **pymango**
- Créer des requêtes simples
- Manipuler des index
- Créer des requêtes d'agrégation et de modification

#### Recommandations clés :

- Révision générale du résumé théorique



**05 heures**

# CONSIGNES

## 1. Pour le formateur :

- Rappeler la syntaxe de base de création et manipulation des documents
- Interroger les documents en ligne de commande

## 2. Pour l'apprenant :

- Manipuler les commandes d'installation de pymango
- Manipuler les commandes de lancement du serveur,
- Ecrire des requêtes en ligne de commande

## 3. Conditions de réalisation :

- Support de résumé théorique accompagnant

## 4. Critères de réussite :

- Le stagiaire est-il capable de :
  - Installer pymango
  - Se connecter au serveur de la base
  - Ecrire des requêtes et manipuler le résultat avec Python



## Activité 2

### Installation et lancement du serveur



#### Installation et configuration

1. Dans un terminal (invite de commande), lancer l'installation de **pymongo**,
2. Utiliser **MongoClient** pour effectuer une connexion au serveur ,
3. Se connecter à la base de données « **DBSportifs** » déjà élaborée,

## Activité 2

### Installation et lancement du serveur



#### Installation et configuration

1. Dans un terminal (invite de commande), lancer l'installation de **pymongo**,

```
(c) Microsoft Corporation. Tous droits réservés.  
  
C:\Users\<redacted>>pip install pymongo  
Collecting pymongo  
  Downloading pymongo-0.1.1.tar.gz (2.7 kB)  
  Preparing metadata (setup.py) ... done  
Collecting requests>=2.4.3  
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)  
----- 62.8/62.8 kB 281.3 kB/s eta 0:00:00  
Collecting charset-normalizer<3,>=2  
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)  
Collecting certifi>=2017.4.17  
  Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)  
----- 161.1/161.1 kB 802.6 kB/s eta 0:00:00  
Collecting idna<4,>=2.5  
  Downloading idna-3.4-py3-none-any.whl (61 kB)  
----- 61.5/61.5 kB 172.7 kB/s eta 0:00:00  
Collecting urllib3<1.27,>=1.21.1  
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)  
----- 140.4/140.4 kB 277.5 kB/s eta 0:00:00  
Using legacy 'setup.py install' for pymongo, since package 'wheel' is not installed.  
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests, pymongo  
  Running setup.py install for pymongo ... done  
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 pymongo-0.1.1 requests-2.28.1 urllib3-1.26.12
```

## Activité 2

### Installation et lancement du serveur



#### Installation et configuration

2. Utiliser **MongoClient** pour effectuer une connexion au serveur
3. Se connecter à la base de données « **DBSportifs** » déjà élaborée

```
C:\Users\<redacted>>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from pymongo import MongoClient
>>> client = MongoClient()
>>> maBase = client["DBSportifs"]
>>>
```

## Activité 2

### Interrogation des documents



#### Interrogation des documents

1. Afficher les noms et prénoms des sportifs de la collection
2. Récupérer la moyenne des médailles par sport
3. Supprimer les sportifs du sport « athlétisme »
4. Mettre à jour le nombre des médailles du sportif d'identifiant « sp5 »

## Activité 2

### Interrogation des documents



#### Interrogation des documents

1. Afficher les noms et prénoms des sportifs de la collection

```
resultat = maBase.Sportif.find({}, {"nom":1, "prenom":1})
```

```
for i in resultat[:]:
```

```
    print(i)
```

## Activité 2

### Interrogation des documents



### Interrogation des documents

- Récupérer la moyenne des médailles par sport

```
resultat = maBase.Sportifs.aggregate([\n    {"$group":\n        {"_id": "$sport.description",\n         "moyenne": {"$avg": "$nbMedailles"}}\n    ])\nfor i in resultat:\n    print(i["nbMedailles"], "nombre de medailles", i["_id"])
```





## Activité 3

### Sécuriser une base de données MongoDB

#### Compétences visées :

- Importer / exporter les données
- Sécuriser les accès (authentification )

#### Recommandations clés :

- Révision générale du résumé théorique



05 heures

# CONSIGNES

## 1. Pour le formateur :

- Rappeler la syntaxe de base de importer une base de données en JSON
- Rappeler la syntaxe de base d'exporter une base de données
- Sécuriser l'accès à la base de données

## 2. Pour l'apprenant :

- Manipuler les commandes d'import/export de la base de données,
- Créer des utilisateurs de la base de données
- Supprimer des utilisateurs

## 3. Conditions de réalisation :

- Support de résumé théorique accompagnant

## 4. Critères de réussite :

- Le stagiaire est-il capable de :
  - Importer une base de données mongodb à partir d'un fichier JSON
  - Exporter une base de données mongodb sous forme d'un fichier JSON
  - Créer des utilisateurs et des rôles
  - Supprimer des utilisateurs



## Activité 3

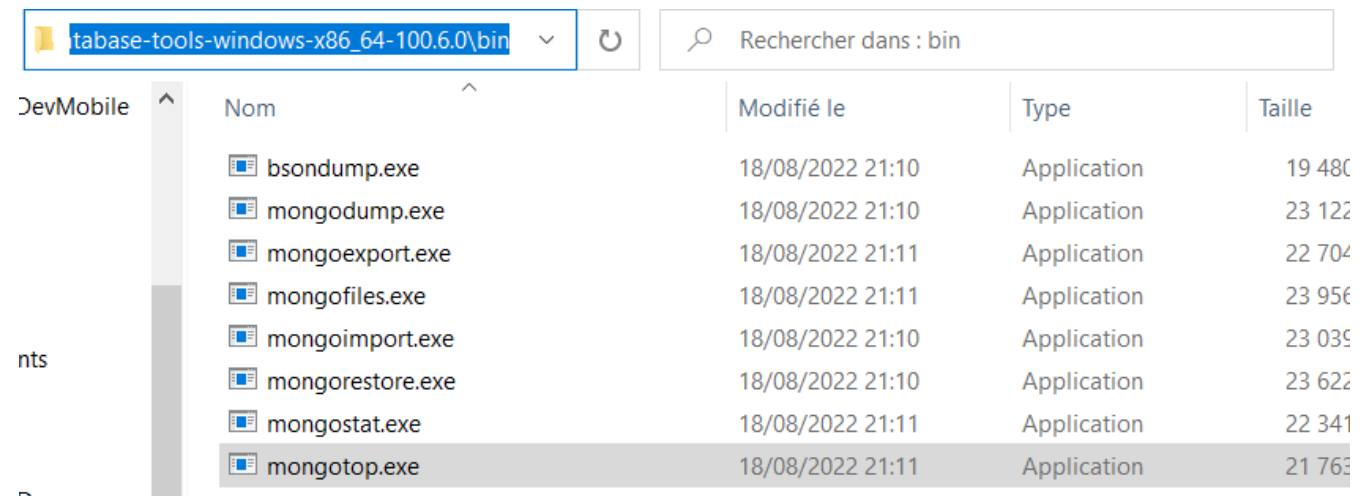
### Exporter/importer une base de données

#### Configuration

1. Pour importer ou exporter une base de données **mongodb** il faut utiliser les programmes **mongoimport** et **mongoexport** qui sont pas installés par **default** avec **mongodb**,
2. Commences par télécharger l'outil **mongodb database tools** à partir de l'Url :

<https://www.mongodb.com/try/download/database-tools>

1. Copier les fichiers du dossier bin dans le dossier bin de **mongodb**


















Nom	Modifié le	Type	Taille
bsondump.exe	18/08/2022 21:10	Application	19 480
mongodump.exe	18/08/2022 21:10	Application	23 122
mongoexport.exe	18/08/2022 21:11	Application	22 704
mongofiles.exe	18/08/2022 21:11	Application	23 956
mongoimport.exe	18/08/2022 21:10	Application	23 039
mongorestore.exe	18/08/2022 21:10	Application	23 622
mongostat.exe	18/08/2022 21:11	Application	22 341
mongotop.exe	18/08/2022 21:11	Application	21 763

## Activité 3

### Exporter/importer une base de données

#### Configuration

C:\Program Files\MongoDB\Server\5.0\bin

Nom	Modifié le	Type	Taille
 bsondump.exe	18/08/2022 21:10	Application	19 480 Ko
 InstallCompass.ps1	03/08/2021 20:22	Script Windows Po...	2 Ko
 mongo.exe	03/08/2021 21:51	Application	21 646 Ko
 mongod.cfg	31/10/2022 21:48	Fichier source Con...	1 Ko
 mongod.exe	03/08/2021 21:51	Application	45 745 Ko
 mongod.pdb	03/08/2021 21:51	VisualStudio.pdb.c...	516 908 Ko
 mongodump.exe	18/08/2022 21:10	Application	23 122 Ko
 mongoexport.exe	18/08/2022 21:11	Application	22 704 Ko
 mongofiles.exe	18/08/2022 21:11	Application	23 956 Ko
 mongoimport.exe	18/08/2022 21:10	Application	23 039 Ko
 mongorestore.exe	18/08/2022 21:10	Application	23 622 Ko
 mongos.exe	03/08/2021 21:11	Application	29 047 Ko
 mongos.pdb	03/08/2021 21:11	VisualStudio.pdb.c...	304 668 Ko
 mongostat.exe	18/08/2022 21:11	Application	22 341 Ko
 mongotop.exe	18/08/2022 21:11	Application	21 763 Ko

## Activité 3

### Exporter/importer une base de données



#### Exporter une base de données MongoDB vers un fichier json

- Ajouter le chemin du dossier bin de **MongoDB** à la variable d'environnement système **Path**
- Ouvrir l'invité de commande et exporter la base de données **DBSportifs** en un fichier **db.json**, en tapant:

```
mongoexport -d DBSportifs -c Sportifs -o dbSportifs.json
```

- Ouvrir le fichier **dbSportifs.json**

```
dbSportifs.json 1 x
D: > {} dbSportifs.json > ...
1 [{"_id":"sp1","nom":"Radi","prenom":"Abdessalam","genre":"homme","sport":{"description":"marathon","olympique":true}}]
2 [{"_id":"sp2","nom":"Larbi","prenom":"Benmbarek","genre":"homme","sport":{"description":"football","olympique":true}}]
3 [{"_id":"sp3","nom":"ELGourch","prenom":"Mohamed","genre":"homme","sport":{"description":"cyclisme","olympique":true}}]
4 [{"_id":"sp4","nom":"Bidouane","prenom":"Nezha","genre":"femme","sport":{"description":"athletisme","olympique":true}}]
5 [{"_id":"sp12","nom":"Rabii","prenom":"Mohamed","genre":"homme","sport":{"description":"box","olympique":true}}]
6 [{"_id":"sp5","nom":"elguerrouj","prenom":"Hicham","genre":"homme","sport":{"description":"box","olympique":true}}]
7 [{"_id":"sp6","nom":"Abissourour","prenom":"Sara","genre":"femme","sport":{"description":"volley ball","olympique":true}}]
8 [{"_id":"sp9","nom":"Belafrikh","prenom":"Amine","genre":"homme","sport":{"description":"Muay Thai","olympique":true}}]
9 [{"_id":"sp10","nom":"Moutawakil","prenom":"Naoual","genre":"femme","nbMedailles":4.0,"sport":{"description":"athletisme","olympique":true}}]
10 [{"_id":"sp11","nom":"Aouita","prenom":"Said","genre":"homme","nbMedailles":3.0,"sport":{"description":"athletisme","olympique":true}}]
```

## Activité 3

### Exporter/importer une base de données

#### Importer une base de données MongoDB à partir d'un fichier Json

- Télécharger le fichier **DBLP.json** du lien suivant : <http://b3d.bdpedia.fr/files/dblp.json.zip>
- Il s'agit d'une base de données bibliographique sous format **json**
- Ouvrir l'invité de commande et importer la base de données **DBLP**

```
mongoimport --host localhost:27017 --db DBLP --collection publis --jsonArray --type json --file dblp.json
```

- Ouvrir une autre instance de la ligne de commande et lancer **mongod.exe**
- Tester l'existence de la base de données **DBLP** en exécutant **show dbs**

```
> show dbs
DBLP      0.001GB
DBSportifs 0.000GB
admin     0.000GB
config    0.000GB
dbJson    0.000GB
local     0.000GB
>
```

## Activité 3

### Création des utilisateurs



#### Création d'un utilisateur de la base de données

1. Créer un utilisateur avec :
  - Le login : « **manager** »
  - Le mot de passe : « **r@@t** »
  - Et le rôle : « **dbAdmin** »
2. Afficher le détail de cet utilisateur

## Activité 3

### Création des utilisateurs



#### Création d'un utilisateur de la base de données

```
use DBSportifs
db.createUser(
  {user: "manager",
    pwd: "r@@t" ,
    "roles" : [
      {
        "role" : "dbAdmin",
        "db" : " DBSportifs "
      }
    ]
  }
)
```

```
> db.createUser({user:"manager","pwd":"r@@t","roles":[{"role":"dbAdmin","db":"DBSportifs"}]})
Successfully added user: {
  "user" : "manager",
  "roles" : [
    {
      "role" : "dbAdmin",
      "db" : "DBSportifs"
    }
  ]
}
```



## Activité 3

### Création des utilisateurs



#### Création d'un utilisateur de la base de données

2. Afficher le détail de cet utilisateur

```
> db.getUser("manager")
{
  "_id" : "test.manager",
  "userId" : UUID("c35bc506-87c5-4672-9f50-07bb9fff47bf"),
  "user" : "manager",
  "db" : "test",
  "roles" : [
    {
      "role" : "dbAdmin",
      "db" : "DBSportifs"
    }
  ],
  "mechanisms" : [
    "SCRAM-SHA-1",
    "SCRAM-SHA-256"
  ]
}
```

## Activité 3

### Création des utilisateurs



#### Authentification à la base de données

- Authentifier vous à la base de données par le compte manager qu'on vient de créer

```
> db.auth("manager","r@t")
1
>
```