

Partie 1 :

1. **(1pt)** Le type de données char réserve autant d'espace qu'on mentionne dans la taille du champs même si la chaîne introduite occupe moins que cette taille l'espace est comme même réservé alors que varchar, n'enregistre que les caractères introduits donc il est plus économique en terme d'espace.
2. **(1pt)** La différence est que dans la première on ramène tous les champs alors que dans la seconde on ne ramène que deux et d'un autre côté il vaut mieux spécifier les champs dans le select car le * prends beaucoup de temps d'exécution.
3. **(1pt)**
 - a. Les avantages :
 - i. Focaliser les données pour un utilisateur
 - ii. Masquer la complexité d'une base de données
 - iii. Simplifier la gestion des droits d'accès des utilisateurs
 - iv. Améliorer la performance
 - v. Organiser les données pour l'exportation vers d'autres applications
 - b. Les inconvénients
 - i. Limitation au niveau des mise à jour.
4. **(1pt)** Le catalogue d'une base de données est la base de données système qui contient toutes les informations concernant les autres bases de données du sgbd ainsi que les informations d'environnement du système. Exemple mysql : la base mysql et la base information_schema.

Partie 2 :**Remarques :**

Les contraintes 2 et 3.C ont été enlevées ainsi que l'état de sortie 3.

A. Création des tables et de la base de données

//**(1pt)** Création de la base de données

```
DROP DATABASE IF EXISTS emd3;  
CREATE DATABASE emd3;  
USE emd3;
```

//**(1pt)** création de la table Candidat

```
DROP TABLE IF EXISTS emd3.candidat;  
CREATE TABLE candidat (  
    id int(11) NOT NULL auto_increment,  
    nom varchar(30) default NULL,  
    pnom varchar(30) default NULL,  
    age tinyint(4) default 0,  
    poids tinyint(4) default 0,  
    wilaya varchar(2) default NULL,  
    PRIMARY KEY (id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

//**(1pt)** création de la table Plat

```
DROP TABLE IF EXISTS emd3.plat;  
CREATE TABLE plat (  
    id int(11) NOT NULL auto_increment,  
    nom varchar(40) default NULL,  
    qtestk int(11) default 0,  
    PRIMARY KEY (id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

//**(1,5pt)** création de la table rencontre

```
DROP TABLE IF EXISTS emd3.rencontre;  
CREATE TABLE rencontre (  
    cnd1 int(11) NOT NULL,  
    cnd2 int(11) NOT NULL,  
    dater date NOT NULL,  
    niveau varchar(30),  
    qtecnd1 int(11) default 0,  
    qtecnd2 int(11) default 0,  
    PRIMARY KEY (cnd1,cnd2,dater),  
    FOREIGN KEY (cnd1) REFERENCES candidat(id) ON DELETE CASCADE ON UPDATE  
CASCADE,  
    FOREIGN KEY (cnd2) REFERENCES candidat(id) ON DELETE CASCADE ON UPDATE  
CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

//**(2pt)** création de la table manger

```
DROP TABLE IF EXISTS emd3.manger;  
CREATE TABLE manger (  
    cnd1 int(11) NOT NULL,  
    cnd2 int(11) NOT NULL,  
    cndm int(11) NOT NULL,  
    idplat int(11) NOT NULL,  
    qteplat int(11) default 0,  
    PRIMARY KEY (cnd1,cnd2,cndm,idplat),  
    FOREIGN KEY (cnd1,cnd2) REFERENCES rencontre(cnd1,cnd2) ON DELETE CASCADE  
ON UPDATE CASCADE,  
    FOREIGN KEY (idplat) REFERENCES Plat(id) ON DELETE CASCADE ON UPDATE  
CASCADE,  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

B. Création des triggers

// Les trois triggers suivants permettent de gérer la quantité en stock du plat ainsi que les quantité consommées dans la rencontre par les deux candidats

(1,5pt)

```
Create trigger emd3.InsQteStk AFTER INSERT on emd3.manger
for each row
BEGIN
    UPDATE plat SET qtestk=qtestk-new.qtePlat WHERE id=new.idPlat;
    IF new.cndm=new.cnd1 THEN
        UPDATE rencontre Set QteCnd1=QteCnd1+new.QtePlat Where cnd1=new.cnd1 And
        cnd2=new.cnd2;
    ELSE
        UPDATE rencontre Set QteCnd2=QteCnd2+new.QtePlat Where cnd1=new.cnd1 And
        cnd2=new.cnd2;
    END IF;
END
```

(1,5pt)

```
Create trigger emd3.UpdQteStk AFTER UPDATE on emd3.manger
for each row
BEGIN
    IF (new.qtePlat<>old.qtePlat) THEN
        UPDATE plat SET qtestk=qtestk+old.qtePlat-new.qtePlat WHERE
        id=new.idPlat;
        IF old.cndm=old.cnd1 THEN
            UPDATE rencontre Set QteCnd1=QteCnd1-old.QtePlat+new.qtePlat Where
            cnd1=new.cnd1 And cnd2=new.cnd2;
        ELSE
            UPDATE rencontre Set QteCnd2=QteCnd2-old.QtePlat+new.qtePlat Where
            cnd1=new.cnd1 And cnd2=new.cnd2;
        END IF;
    END IF;
END
```

(1,5pt)

```
Create trigger emd3.DelQteStk AFTER DELETE on emd3.manger
for each row
BEGIN
    UPDATE plat SET qtestk=qtestk+old.qtePlat WHERE id=old.idPlat;
    IF old.cndm=old.cnd1 THEN
        UPDATE rencontre Set QteCnd1=QteCnd1-old.QtePlat Where cnd1=old.cnd1 And
        cnd2=old.cnd2;
    ELSE
        UPDATE rencontre Set QteCnd2=QteCnd2-old.QtePlat Where cnd1=old.cnd1 And
        cnd2=old.cnd2;
    END IF;
END
```

C. Création des procédures stockées : On a besoin ici d'une procédure stockée pour l'insertion et la modification dans toutes les tables de la base, mais on se contente de présenter uniquement la plus délicate qui concerne l'ajout d'une rencontre car c'est là où on doit effectuer les vérification des contraintes et des règles de gestion :

(2,5pt)

```
DELIMITER $$
CREATE PROCEDURE new_rencontre(cnd1 int,cnd2 int,dater date,niveau int, OUT
result INTEGER(11))
BEGIN
    //vérification de la contrainte 1 et 2.
    IF NOT EXISTS (
        Select  c1.id
        From      candidat C1, candidat C2
        Where      (abs(c1.age-c2.age)>2 Or
                    abs(c1.poids-c2.poids)>5) And
                    c1.id=cnd1 AND
                    c2.id=cnd2
    ) THEN
        //Vérification de la contrainte 4.
        IF NOT EXISTS(
            Select  count(cnd1)
            From      rencontre r
            Where      (r.cnd1=cnd1 OR
                        r.cnd2=cnd1) AND
                        r.dater=dater
            HAVING count(cnd1)>3
        ) THEN
            IF NOT EXISTS(
                Select  count(cnd1)
                From      rencontre r
                Where      r.cnd1=cnd2 OR
                        r.cnd2=cnd2 AND
                        r.dater=dater
                HAVING count(cnd1)>3
            ) THEN
                //vérifier que cnd1 et cnd2 sont différents
                IF cnd1<>cnd2 THEN
                    INSERT INTO rencontre
                    values(cnd1,cnd2,dater,niveau,0,0);
                    SET result = 1;
                Else
                    SET result = -1;
                End IF;
            Else
                SET result = -1;
            End IF;
        Else
            SET result = -1;
        End IF;
    Else
        SET result = -1;
    End IF;
END $$
DELIMITER ;
```

D. Création des vues :**(1pt)**

Create view gagnant

As

```
Select cnd1 gagnant, cnd2 perdant
From rencontre
Where qtecnd1>qtecnd2
Union
Select cnd2 gagnant, cnd1 perdant
From rencontre
Where qtecnd2>qtecnd1;
```

(1,5pt)

Create view etat1

As

```
Select DISTINCT c1.*
From
    candidat C1,
    candidat C2,
    gagnant G
Where    c1.id=g.gagnant and
        c2.id=g.perdant and
        c1.age<c2.age and
        c1.poids<c2.poids;
```

(1,5pt)

Create View Etat2

As

```
Select c.*
From candidat c join gagnant g on g.gagnant=c.id
Where    c.age=(
            Select Min(age)
            From Candidat)
Having count(g.gagnant)>3;
```