Université de Carthage Institut Préparatoire aux Etudes d'Ingénieur de Nabeul Département : Mathématiques	IPEIN المعهد التحضيري للدراسات الهندسية بنابل Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul	Année universitaire : 2020/2021 Filière : SM/SP/ST Niveau d'étude : 2ème année Semestre : 2 Nombre de pages : 2 pages Date : 11/03/2021, Durée : 1h30

DS Informatique N° 2 ~ corrigé v1.0

Exercice 1 : [11 pts]

1) 2 points

```
#1.a
import sqlite3
#1.b
con = sqlite3.connect("concours.db")
#1.c
requete = """
CREATE TABLE "NOTATION" (
    `CodeEpreuve`   NUMERIC NOT NULL,
    `NumCandidat`   NUMERIC NOT NULL,
    `note`  NUMERIC NOT NULL CHECK(note>=0 and note<=20),
    PRIMARY KEY(CodeEpreuve, NumCandidat),
    FOREIGN KEY(`CodeEpreuve`) REFERENCES EPREUVE,
    FOREIGN KEY(`NumCandidat`) REFERENCES CANDIDAT
);
"""
con.execute(requete)

#1.d
con.commit()
con.close()
```

2) 1 point

```
INSERT INTO `NOTATION`(`CodeEpreuve`,`NumCandidat`,`note`) VALUES (3,101,18.5)
#Autrement
INSERT INTO `NOTATION` VALUES (3,101,18.5)
```

3) 1 point

```
UPDATE      NOTATION
SET         note = 17
WHERE       NumCandidat = 101 AND CodeEpreuve = 2;
```

4) 1 point

```
SELECT      NumCandidat
FROM        NOTATION
WHERE       CodeEpreuve = 4 AND note < 8
```

5) 1 point

```
1. SELECT     NumCandidat , note
2. FROM       NOTATION
3. WHERE      CodeEpreuve = 3
4. ORDER BY   note DESC
```

6) 1 point

```
1. SELECT     NomCandidat , note
2. FROM       NOTATION  JOIN    CANDIDAT
3. ON         NOTATION.NumCandidat = CANDIDAT.NumCandidat
4. WHERE      CodeEpreuve =3
```

7) 1 point

```
1. SELECT     AVG (note)
2. FROM       EPREUVE  JOIN   NOTATION
3. ON         EPREUVE.CodeEpreuve = NOTATION.CodeEpreuve
4. WHERE      DesignEpreuve = 'philosophie'
```

8) 1 point

```
1. SELECT     DesignEpreuve , AVG(note) , MIN(note) ,MAX(note)
2. FROM       EPREUVE   JOIN    NOTATION
3. ON         EPREUVE.CodeEpreuve = NOTATION.CodeEpreuve
4. GROUP BY   DesignEpreuve
```

9) 1 point

```
1. SELECT     SUM(N.note * E.Coeff) / SUM(E.Coeff) AS 'Moyenne'
2. FROM       EPREUVE E, CANDIDAT C, NOTATION N
3. WHERE      E.CodeEpreuve = N.CodeEpreuve
4. AND        C.NumCandidat = N.NumCandidat
5. GROUP BY   C.NumCandidat
6. HAVING     Moyenne >= 10
```

10) 1 point

```
1. SELECT NumCandidat
2. FROM   NOTATION
3. WHERE  CodeEpreuve =1
4. AND    note >= ( SELECT AVG(note) FROM NOTATION WHERE CodeEpreuve =1 )
```

Exercice 2: [9 pts]

1) 1 point

Réponse :

Dans la table covid, aucun attribut n'a une valeur unique par entrée. Aucun attribut de peut donc servir de clé primaire.

Si l'on imagine qu'il n'y a, dans cette table, qu'une entrée par année et par pays, le couple (nom, annee) mais aussi le couple (iso, annee) peut servir de clé primaire.

2) 1 point

```sql
SELECT iso, (cas * 100 000 / pop) AS incidence
FROM   covid  JOIN  demographie  ON      iso = pays AND periode = annee
WHERE  annee = 2020
```

3) 1 point

```sql
SELECT nom FROM covid WHERE annee = 2020
AND    deces = (SELECT MAX(deces) FROM covid)
```

4) 1 point

```sql
SELECT      nom
FROM        covid
WHERE       annee = 2021
ORDER BY    cas   DESC  LIMIT 1 OFFSET 1
```

5.a) 1 point

```python
def decesParPays (annee):
    assert  type(annee) == int
     requete = """
                SELECT nom, deces
                FROM   covid
                WHERE  annee = ?
                """
    cur = con.cursor()
    cur.execute(requete,(annee))
    return cur.fetchAll()
```

5.b) 2 points

```
>>> deces2021
[('ITALY', 27227), ('TUNISIA', 1521), ('ALGERIA', 1750)]
```

5.c) 2 points

```python
def tri_pays(L):
    return sorted(L, key = lambda pays : pays[1])

#test
deces2021 = [('ITALY', 27227), ('TUNISIA', 1521), ('ALGERIA', 1 750)]

tri_chaine(deces2021)
[('TUNISIA', 1521), ('ALGERIA', 1 750), ('ITALY', 27227)]
```