SGBD TP1

Exercice 1

- Insérer les avions suivants dans la table Avion: (100, AIRBUS, 300, PARIS), (101,B737,250,NICE), (101, B737,220,PARIS)
  Lors de l'ajout de 3ième Avion , j'ai remarqué que la clé primaire existe déja, donc il n'a pas pu être ajouté ainsi j'ai changé en NA 101 en 102

```
CREATE TABLE Avion( NA INT PRIMARY KEY, Nom
Varchar(12),Capacite INT,Localite
Varchar(10));

CREATE TABLE Pilote ( NP INT, Nom
Varchar(25), Adresse Varchar(40));

CREATE TABLE Vol(NV Varchar(6), NP INT, NA
INT, VD Varchar(10), Va Varchar(10), HD INT,
HA INT);

INSERT INTO Avion (NA,Nom,Capacite,Localite)
values (100,'AIRBUS',300,'PARIS');
INSERT INTO Avion (NA,Nom,Capacite,Localite)
values (101,'B737',250,'NICE');
```

```
INSERT INTO Avion (NA,Nom,Capacite,Localite)
values (102,'B737',220,'PARIS');
```

- Afficher tous les avions

```
SELECT * FROM Avion;
```

- Afficher tous les avions par ordre croissant sur le nom

```
SELECT Nom FROM Avion ORDER BY Nom;
+--------+
| Nom    |
+--------+
| AIRBUS |
| B737   |
| B737   |
+--------+
```

Order By permet de les ordonner par nom et alphabétiquement croissant

- Afficher les noms et les capacités des avions

```
SELECT Nom,Capacite FROM Avion;
+--------+----------+
```

```
| Nom    | Capacite |
+--------+----------+
| AIRBUS |      300 |
| B737   |      250 |
| B737   |      220 |
+--------+----------+
```

- Afficher les localités des avions sans redondance

```
SELECT distinct Localite FROM Avion;
+----------+
| Localite |
+----------+
| PARIS    |
| NICE     |
+----------+
```

Le mot clé 'distinct' permet d'éviter la redondance.

- Afficher les avions dans la localité et PARIS ou NICE

```
SELECT  Localite FROM Avion where Localite =
'Paris' OR Localite = 'Nice';
+----------+
| Localite |
```

```
+----------+
| PARIS    |
| NICE     |
| PARIS    |
+----------+
```

- Modifier la capacité de l'avion numéro 101, la nouvelle capacité et 220

```
UPDATE Avion SET Capacite=220 WHERE NA=101;

SELECT * FROM Avion;
+-----+--------+----------+----------+
| NA  | Nom    | Capacite | Localite |
+-----+--------+----------+----------+
| 100 | AIRBUS |      300 | PARIS    |
| 101 | B737   |      220 | NICE     |
| 102 | B737   |      220 | PARIS    |
+-----+--------+----------+----------+
```

Le mot clé 'UPDATE' permet de mettre à jour notre table combiné avec 'SET' qui permet de modifier tous les avions avec le numéro '101'

- Supprimer les avions dans la capacité et inférieure à 200

```
DELETE FROM Avion WHERE Capacite < 200;
SELECT * FROM Avion ;
+-----+--------+----------+----------+
| NA  | Nom    | Capacite | Localite |
+-----+--------+----------+----------+
| 100 | AIRBUS |      300 | PARIS    |
| 101 | B737   |      220 | NICE     |
| 102 | B737   |      220 | PARIS    |
+-----+--------+----------+----------+
```

Le mot clé 'DELETE' permet d'enlever les lignes du tableau dont la capacité est strictement inférieur à 200

- Afficher la capacité maximale, minimale, moyenne des avions

```
SELECT
MAX(Capacite),MIN(Capacite),AVG(Capacite)
FROM Avion;
+---------------+---------------+---------------+
| MAX(Capacite) | MIN(Capacite) |
AVG(Capacite) |
+---------------+---------------+----------+
|           300 |           220 |       246.6667 |
+---------+-----------+--------------+
```

On utilise les fonctions agrégées qui calcule automatiquement le maximum 'MAX' , minimum 'MIN' et La moyenne 'AVG' de l'avion ici
et on les affiche.

- Afficher les données des avions dont la capacité et la plus basse

```
SELECT * FROM Avion WHERE Capacite = (SELECT
MIN(Capacite) FROM Avion);
+-----+------+----------+----------+
| NA  | Nom  | Capacite | Localite |
+-----+------+----------+----------+
| 101 | B737 |      220 | NICE     |
| 102 | B737 |      220 | PARIS    |
+-----+------+----------+----------+
```

On doit faire une sous requête ici afin de pouvoir comparer avec la capacité la plus basse dans La table Avion.

- Afficher les données des avions dont la capacité et supérieure à la capacité moyenne

```
SELECT * FROM Avion WHERE Capacite > (SELECT
AVG(Capacite) FROM Avion);
```

```
+-----+--------+----------+----------+
| NA  | Nom    | Capacite | Localite |
+-----+--------+----------+----------+
| 100 | AIRBUS |      300 | PARIS    |
+-----+--------+----------+----------+
```

On utilise la même procédure que l'exercice précédent

```
INSERT INTO Pilote (NP, Nom, Adresse) VALUES
(1, 'John Doe', '123 Rue de Paris, Paris');
INSERT INTO Pilote (NP, Nom, Adresse) VALUES
(2, 'Jane Smith', '456 Boulevard, Nice');
INSERT INTO Pilote (NP, Nom, Adresse) VALUES
(3, 'Alice Johnson', '789 Avenue de Lyon,
Lyon');


INSERT INTO Vol (NV, NP, NA, VD, Va, HD, HA)
VALUES ('IT100', 1, 100, 'Paris', 'Nice', 10,
12);
INSERT INTO Vol (NV, NP, NA, VD, Va, HD, HA)
VALUES ('IT101', 2, 101, 'Nice', 'Paris', 15,
17);
INSERT INTO Vol (NV, NP, NA, VD, Va, HD, HA)
VALUES ('IT102', 3, 102, 'Paris', 'Lyon', 18,
20);
INSERT INTO Vol (NV, NP, NA, VD, Va, HD, HA)
```

```
VALUES ('IT104', 1, 100, 'PARIS', 'NICE', 9,
11);
```

Ici nous remplissons notre base de données afin de tester nos requêtes.

- Afficher le nom et l'adresse des pilotes assurant les vols IT100 et IT104

```
SELECT distinct Nom, Adresse FROM Pilote, Vol
WHERE Pilote.NP = Vol.Np AND (Vol.NV =
'IT100' OR Vol.NV= 'IT104');
+----------+--------------------------+
| Nom      | Adresse                  |
+----------+--------------------------+
| John Doe | 123 Rue de Paris, Paris  |
+----------+--------------------------+
```

Nous avons bien notre pilote qui assure le vol IT100 et IT104, nous avons ajouté le mot clé distinct afin de pas avoir de doublons.

- Afficher les numéros des pilotes qui sont en service

```
INSERT INTO Pilote (NP, Nom, Adresse) VALUES
(4, 'Michael Brown', '321 Rue de Lyon,
Lyon'); INSERT INTO Pilote (NP, Nom, Adresse)
```

```
VALUES (5, 'David Wilson', '654 Avenue de
Bordeaux, Bordeaux');

SELECT distinct Pilote.NP FROM Pilote, Vol
WHERE Pilote.NP = Vol.Np;
+------+
| NP   |
+------+
|    1 |
|    2 |
|    3 |
+------+
```

J'ai ajouté des pilotes qui n'ont pas de vol attribué afin de tester.
J'ai choisi de pas utiliser la jointure ici même si on pouvais afin de simplifier la requête.

- Afficher les numéros des pilotes qui ne sont pas en service

```
SELECT DISTINCT Pilote.NP FROM Pilote LEFT
JOIN Vol ON Pilote.NP = Vol.NP WHERE Vol.NP
IS NULL;
+------+
| NP   |
+------+
```

```
|    4 |
|    5 |
+------+
```

On utilise ici la jointure, sans le left JOIN on aurais des resultats erronés.

- Afficher les noms des pilotes qui conduisent un AIRBUS

```
SELECT distinct Pilote.Nom FROM
Pilote,Vol,Avion WHERE Pilote.NP = Vol.NP AND
Avion.Nom = 'AIRBUS' AND Avion.NA = Vol.NA;
+----------+
| Nom      |
+----------+
| John Doe |
+----------+
```

Ici on utilise distinct afin d'éviter les doublons.
Le reste de la commande est logique

Exercice 2:

1- créer la bd correspondante.

```
CREATE DATABASE GestionArticles;
USE GestionArticles;
ARTICLE CREATE TABLE ARTICLE ( NoArt INT
PRIMARY KEY, Libelle VARCHAR(50), Stock INT
CHECK (Stock > 0) );
CREATE TABLE FOURNISSEUR ( NoFour INT PRIMARY
KEY, NomF VARCHAR(50) NOT NULL, Adresse
VARCHAR(100), VilleFour VARCHAR(50) );
CREATE TABLE FOURNIR ( NoFour INT, NoArt INT,
PrixArticle DECIMAL(10, 2) CHECK (PrixArticle
> 0), Delai INT DEFAULT 2, PRIMARY KEY
(NoFour, NoArt),
FOREIGN KEY (NoFour) REFERENCES
FOURNISSEUR(NoFour),
FOREIGN KEY (NoArt) REFERENCES ARTICLE(NoArt)
);
```

2- remplir les tables par des exemples.

```
INSERT INTO ARTICLE (NoArt, Libelle, Stock)
VALUES
(1, 'Chaise', 100),
(2, 'Table', 50),
(3, 'Lampe', 200),
(4, 'Fauteuil', 3);
INSERT INTO FOURNISSEUR (NoFour, NomF,
Adresse, VilleFour) VALUES
(1, 'Fournisseur A', '123 Rue des
```

```
Fournisseurs', 'Paris'),
(2, 'Fournisseur B', '456 Boulevard de
Paris', 'Lyon'),
(3, 'Fournisseur C', '789 Avenue de
Toulouse', 'Marseille'),
(4, 'Particulier A', '1011 Avenue de
Vitry','Vitry-sur-Seine');

INSERT INTO FOURNIR (NoFour, NoArt,
PrixArticle, Delai) VALUES
(1, 1, 15.50, 3),
(2, 2, 30.00, 2),
(1, 3, 10.75, 4),
(3, 1, 18.00, 1),
(3, 4, 135.00, 30);
```

3– donner les numéros et les libellés des articles de stock inférieur à 10.

```
SELECT NoART FROM ARTICLE WHERE Stock<10;
+-------+
| NoART |
+-------+
|     4 |
+-------+
```

4- afficher la liste des articles dont le prix est compris entre 100 et 300

```
SELECT ARTICLE.Libelle
FROM ARTICLE
JOIN FOURNIR
ON ARTICLE.NoArt = FOURNIR.NoArt
WHERE FOURNIR.PrixArticle > 100 AND
FOURNIR.PrixArticle < 300;
+----------+
| Libelle  |
+----------+
| Fauteuil |
+----------+
```

Utilisation classique de JOIN afin d'afficher les articles dans la tranche de prix indiqué

5- lister les fournisseurs dont le nom commence par “Fo” .

```
SELECT NomF
FROM FOURNISSEUR
WHERE NomF LIKE 'Fo%';
+---------------+
| NomF          |
+---------------+
| Fournisseur A |
| Fournisseur B |
| Fournisseur C |
```

```
+--------------+
```

On utilise le mot clé 'LIKE' pour trouver des mot commencant par 'Fo' suivi de n'importe quels caractères grace au symbole '%'

6- donner les noms et les adresses des fournisseurs qui proposent des articles pour lesquels le délai d'approvisionnement est supérieur à 20 jours.

```
UPDATE FOURNIR SET Delai = 30 WHERE NoArt = 4
AND NoFour = 3;

SELECT NomF, Adresse
FROM FOURNISSEUR
JOIN FOURNIR ON FOURNISSEUR.NoFour=
FOURNIR.NoFour
WHERE Delai > 20;
+---------------+--------------------------+
| NomF          | Adresse                  |
+---------------+--------------------------+
| Fournisseur C | 789 Avenue de Toulouse |
+---------------+--------------------------+
```

J'ai changé le delai de livraison du fauteuil pour qu'on puisse vérifier le resultat.

Utilisation de join et avec la correspondance de NoFour en commun

7- afficher la liste pour chaque article (numéro et libellé) qui a du prix d'achat maximum, minimum ou moyen.

```
SELECT ARTICLE.NoArt,
ARTICLE.Libelle, MAX(FOURNIR.PrixArticle) AS
PrixMax, MIN(FOURNIR.PrixArticle) AS PrixMin,
AVG(FOURNIR.PrixArticle) AS PrixMoyen
FROM ARTICLE JOIN FOURNIR ON ARTICLE.NoArt =
FOURNIR.NoArt
GROUP BY ARTICLE.NoArt, ARTICLE.Libelle;
+-------+----------+---------+---------+-----
-------+
| NoArt | Libelle  | PrixMax | PrixMin |
PrixMoyen  |
+-------+----------+---------+---------+-----
-------+
|     1 | Chaise   |   18.00 |   15.50 |
16.750000 |
|     2 | Table    |   30.00 |   30.00 |
30.000000 |
|     3 | Lampe    |   10.75 |   10.75 |
10.750000 |
|     4 | Fauteuil |  135.00 |  135.00 |
135.000000 |
```

```
+-------+----------+---------+---------+-----
-------+
```

Ici vu qu'on a pas plusieurs fournisseurs avec des articles différents l'Intérêt est moindre ici mais on a bien le prixmax, prixmin et prixmoyen de tous les fournisseurs.

Exercice 3:

```sql
CREATE DATABASE GestionNotes;

USE GestionNotes;

CREATE TABLE ETUDIANT (
    NEtudiant INT PRIMARY KEY,
    Nom VARCHAR(50),
    Prénom VARCHAR(50)
);

CREATE TABLE MATIERE (
    CodeMat INT PRIMARY KEY,
    LibelléMat VARCHAR(100),
    CoeffMat INT CHECK (CoeffMat > 0)
);

CREATE TABLE EVALUER (
```

```
    NEtudiant INT,
    CodeMat INT,
    Date DATE,
    Note DECIMAL(5, 2) CHECK (Note >= 0 AND
Note <= 20),
PRIMARY KEY (NEtudiant, CodeMat),
FOREIGN KEY (NEtudiant) REFERENCES
ETUDIANT(NEtudiant),
FOREIGN KEY (CodeMat) REFERENCES
MATIERE(CodeMat)
);
```

Nous avons crée notre base de données en faisant attention a bien utiliser le mot clé 'CHECK' pour pour délimiter les notes et aussi pour avoir les coefficients > 0 par sécurité.
Nous avons utilisé 'DECIMAL' pour les notes avec 5 entier et 2 chiffres après la virgules.
nous avons aussi utilisé 'DATE' pour le format de date dans la table 'EVALUER'

```
INSERT INTO ETUDIANT (NEtudiant, Nom, Prénom)
VALUES (1, 'Dara', 'Pak'),
       (2, 'Alexis', 'Xiong'),
       (3, 'Hocine', 'AIT YAKOUB'),
       (4, 'Houda', 'Slimani'),
       (5, 'Jérémy', 'Moscato'),
       (6, 'Lomig', 'Nael'),
```

```
        (7, 'mekki', 'mekki'),
        (8, 'Neo', 'Pagliara'),
        (9, 'Remy', 'Xu'),
        (10, 'Selladurai', 'Gowshigan'),
        (11, 'Vivien', 'LAZONE');

INSERT INTO MATIERE (CodeMat, LibelléMat)
VALUES (1001, 'SGBD'),
        (1002, 'TO'),
        (1003, 'ATDN'),
        (1004, 'CAA'),
        (1005, 'Anglais'),
        (1006, 'ANS');

INSERT INTO EVALUER (NEtudiant, CodeMat,
Note)
VALUES (1, 1001, 15),
        (2, 1002, 18),
        (3, 1003, 12),
        (4, 1004, 17),
        (5, 1005, 14),
        (6, 1006, 16),
        (7, 1001, 13),
        (8, 1002, 20),
        (9, 1003, 10),
        (10, 1004, 19),
        (11, 1005, 15);
```

Création de notre base de données .

1 – Calculer le nombre total d'étudiants .

```
SELECT COUNT(*) FROM ETUDIANT;
+----------+
| COUNT(*) |
+----------+
|       11 |
+----------+
```

Nous allons utiliser la fonction agrégée 'COUNT' pour compter nos étudiants

2 – Quelles sont, parmi l'ensemble des notes, la note la plus haute et la note la plus basse ?

```
SELECT MAX(Note) as NoteMaximum,MIN(Note)as
NoteMinimum FROM EVALUER;
+-------------+-------------+
| NoteMaximum | NoteMinimum |
+-------------+-------------+
|       20.00 |       10.00 |
+-------------+-------------+
```

3 – Afficher les moyennes de chaque étudiant dans chacune des matières.

```
INSERT INTO EVALUER (NEtudiant, CodeMat,
Note)
VALUES
(1, 101, 16),
(1, 102, 14),
(1, 103, 13),
(1, 104, 17),
(1, 105, 18),
(1, 106, 12),
(2, 101, 20),
(2, 102, 19),
(2, 103, 18),
(2, 104, 15),
(2, 105, 14),
(2, 106, 10),
(3, 101, 14),
(3, 102, 12),
(3, 103, 10),
(3, 104, 13),
(3, 105, 15),
(3, 106, 16),
(4, 101, 19),
(4, 102, 18),
(4, 103, 17),
(4, 104, 15),
(4, 105, 16),
(4, 106, 14),
(5, 101, 13),
(5, 102, 14),
```

```
(5, 103, 15),
(5, 104, 16),
(5, 105, 12),
(5, 106, 17),
(6, 101, 16),
(6, 102, 17),
(6, 103, 15),
(6, 104, 18),
(6, 105, 14),
(6, 106, 19),
(7, 101, 15),
(7, 102, 14),
(7, 103, 16),
(7, 104, 13),
(7, 105, 14),
(7, 106, 12),
(8, 101, 19),
(8, 102, 20),
(8, 103, 18),
(8, 104, 16),
(8, 105, 14),
(8, 106, 15),
(9, 101, 12),
(9, 102, 13),
(9, 103, 14),
(9, 104, 12),
(9, 105, 10),
(9, 106, 11),
(10, 101, 17),
```

```
(10, 102, 19),
(10, 103, 18),
(10, 104, 20),
(10, 105, 18),
(10, 106, 16),
(11, 101, 16),
(11, 102, 15),
(11, 103, 14),
(11, 104, 16),
(11, 105, 17),
(11, 106, 15);

SELECT
    N.Nom,
    M.LibelléMat,
    AVG(E.Note) AS Moyenne
FROM
    EVALUER E
JOIN
    ETUDIANT N ON E.NEtudiant = N.NEtudiant
JOIN
    MATIERE M ON E.CodeMat = M.CodeMat
GROUP BY
    N.Nom, M.LibelléMat
ORDER BY
    N.Nom, M.LibelléMat;
------------+-------------+-----------+
| Nom        | LibelléMat  | Moyenne   |
+------------+-------------+-----------+
```

Alexis	Anglais	14.000000
Alexis	ANS	10.000000
Alexis	ATDN	18.000000
Alexis	CAA	15.000000
Alexis	SGBD	20.000000
Alexis	TO	18.500000
Dara	Anglais	18.000000
Dara	ANS	12.000000
Dara	ATDN	13.000000
Dara	CAA	17.000000
Dara	SGBD	15.500000
Dara	TO	14.000000
Hocine	Anglais	15.000000
Hocine	ANS	16.000000
Hocine	ATDN	11.000000
Hocine	CAA	13.000000
Hocine	SGBD	14.000000
Hocine	TO	12.000000
Houda	Anglais	16.000000
Houda	ANS	14.000000
Houda	ATDN	17.000000
Houda	CAA	16.000000
Houda	SGBD	19.000000
Houda	TO	18.000000
Jérémy	Anglais	13.000000
Jérémy	ANS	17.000000
Jérémy	ATDN	15.000000
Jérémy	CAA	16.000000
Jérémy	SGBD	13.000000

Jérémy	TO	14.000000
Lomig	Anglais	14.000000
Lomig	ANS	17.500000
Lomig	ATDN	15.000000
Lomig	CAA	18.000000
Lomig	SGBD	16.000000
Lomig	TO	17.000000
mekki	Anglais	14.000000
mekki	ANS	12.000000
mekki	ATDN	16.000000
mekki	CAA	13.000000
mekki	SGBD	14.000000
mekki	TO	14.000000
Neo	Anglais	14.000000
Neo	ANS	15.000000
Neo	ATDN	18.000000
Neo	CAA	16.000000
Neo	SGBD	19.000000
Neo	TO	20.000000
Remy	Anglais	10.000000
Remy	ANS	11.000000
Remy	ATDN	12.000000
Remy	CAA	12.000000
Remy	SGBD	12.000000
Remy	TO	13.000000
Selladurai	Anglais	18.000000
Selladurai	ANS	16.000000
Selladurai	ATDN	18.000000
Selladurai	CAA	19.500000

```
| Selladurai | SGBD         | 17.000000 |
| Selladurai | TO           | 19.000000 |
| Vivien     | Anglais      | 16.000000 |
| Vivien     | ANS          | 15.000000 |
| Vivien     | ATDN         | 14.000000 |
| Vivien     | CAA          | 16.000000 |
| Vivien     | SGBD         | 16.000000 |
| Vivien     | TO           | 15.000000 |
+------------+--------------+-----------+
```

On a ajouté plus de notes pour l'exercices, nous avons affiché pour chaque nom étudiant leur moyenne par matières. Pour cela on a calculé leur moyenne, utilisé la jointure afin de récupérer toutes les informations nécessaire puis groupé et ordonnée par le nom de l'étudiant et la nom de matière.

4 – Quelles sont les moyennes par matière ? On utilisera la requête de la question 3 comme table source

```
SELECT
    LibelléMat,
    AVG(Moyenne) AS MoyenneParMatiere
FROM (
    SELECT
        N.Nom,
```

```
        M.LibelléMat,
        AVG(E.Note) AS Moyenne
    FROM
        EVALUER E
    JOIN
        ETUDIANT N ON E.NEtudiant =
N.NEtudiant
    JOIN
        MATIERE M ON E.CodeMat = M.CodeMat
    GROUP BY
        N.Nom, M.LibelléMat
) AS MoyennesParEtudiant
GROUP BY
    LibelléMat
ORDER BY
    LibelléMat;
+-------------+--------------------+
| LibelléMat  | MoyenneParMatiere  |
+-------------+--------------------+
| Anglais     |     14.7272727273  |
| ANS         |     14.1363636364  |
| ATDN        |     15.1818181818  |
| CAA         |     15.5909090909  |
| SGBD        |     15.9545454545  |
| TO          |     15.8636363636  |
+-------------+--------------------+
```

on a utilisé la requête de la question 3 comme une sous requête afin de chercher la moyenne de la classe pour chaque matière.

5 – Trouver la moyenne générale de chaque étudiant . On utilisera la requête de la question 3 comme table source

```
SELECT
    Nom,
    AVG(Moyenne) AS MoyenneParMatiere
FROM (
    SELECT
        N.Nom,
        M.LibelléMat,
        AVG(E.Note) AS Moyenne
    FROM
        EVALUER E
    JOIN
        ETUDIANT N ON E.NEtudiant =
N.NEtudiant
    JOIN
        MATIERE M ON E.CodeMat = M.CodeMat
    GROUP BY
        N.Nom, M.LibelléMat
) AS MoyennesParEtudiant
GROUP BY
    Nom
```

```
ORDER BY
    Nom;
+------------+------------------+
| Nom        | MoyenneGenerale |
+------------+------------------+
| Alexis     |   15.9166666667 |
| Dara       |   14.9166666667 |
| Hocine     |   13.5000000000 |
| Houda      |   16.6666666667 |
| Jérémy     |   14.6666666667 |
| Lomig      |   16.2500000000 |
| mekki      |   13.8333333333 |
| Neo        |   17.0000000000 |
| Remy       |   11.6666666667 |
| Selladurai |   17.9166666667 |
| Vivien     |   15.3333333333 |
+------------+------------------+
```

On change juste dans le SELECT ce qu'on veut dans ce cas 'Nom' et on groupe et ordonne par 'Nom' afin d'avoir la moyenne par élève.

6 – Quelle est la moyenne générale de la promotion ? On utilisera la requête de la question 5 comme table source

```
SELECT AVG(MoyenneParMatiere) as MoyennePromo
FROM (
SELECT
    Nom,
    AVG(Moyenne) AS MoyenneParMatiere
FROM (
    SELECT
        N.Nom,
        M.LibelléMat,
        AVG(E.Note) AS Moyenne
    FROM
        EVALUER E
    JOIN
        ETUDIANT N ON E.NEtudiant =
N.NEtudiant
    JOIN
        MATIERE M ON E.CodeMat = M.CodeMat
    GROUP BY
        N.Nom, M.LibelléMat
) AS MoyennesParEtudiant
GROUP BY Nom
)As MoyenneParEtudiantTotal;
+--------------------+
| MoyennePromo       |
+--------------------+
| 15.24242424243636 |
+--------------------+
```

On a procédé de la même manière que précédemment on a juste fais la moyenne de tout les notes de tous les élèves afin d'avoir la moyenne de la promotion.

7 – Quels sont les étudiants qui ont une moyenne générale supérieure ou égale à la moyenne générale de la promotion ?On utilisera la requête de la question 5 comme table source (vue)

```
CREATE VIEW MoyennesEtudiants AS
SELECT
    N.Nom,
    AVG(E.Note) AS MoyenneGenerale
FROM
    EVALUER E
JOIN
    ETUDIANT N ON E.NEtudiant = N.NEtudiant
JOIN
    MATIERE M ON E.CodeMat = M.CodeMat
GROUP BY
    N.Nom;

SELECT
    Nom,
    MoyenneGenerale
FROM
    MoyennesEtudiants
```

```
WHERE
    MoyenneGenerale >= (
        SELECT AVG(MoyenneGenerale)
        FROM MoyennesEtudiants
    )
ORDER BY
    Nom;
```

```
+------------+------------------+
| Nom        | MoyenneGenerale  |
+------------+------------------+
| Alexis     |        16.285714 |
| Houda      |        16.571429 |
| Lomig      |        16.428571 |
| Neo        |        17.428571 |
| Selladurai |        18.142857 |
| Vivien     |        15.428571 |
+------------+------------------+
```

on crée une Vue contenant la moyenne des étudiant afin de simplifier la requête suivante permettant d'avoir les élèves qui ont une moyenne supérieure à la moyenne de la promotion.