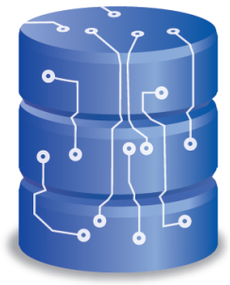

Bases de données

Université Aboubakr Belkaïd de Tlemcen

L3 Génie Industriel

Variables dans SQL

Variables dans SQL

```sql
SELECT
    Etudiant.idEtudiant, nom, note, nomMatiere
FROM
    ETUDIANT,
    MATIERE,
    NOTE
WHERE
    NOTE.idEtudiant = ETUDIANT.idEtudiant
        AND MATIERE.idMatiere = NOTE.idMatiere;
```

160% 48:9

Filter: File:

idEtudiant	nom	note	nomMatiere
1	Samantha	15	Maths
2	Francis	20	Maths
3	Charles	13	Maths
4	Penelope	14	Maths
5	Craig	10	Maths
6	Steve	8	Maths
10	Patricia	5	Maths
1	Samantha	20	Sociologie
10	Patricia	4	Sociologie
11	Patricia	7	Nutrition

Les identifiants, les noms, les notes et les matières respectives des étudiants ayant une note

Variables dans SQL

```sql
SELECT
    E.idEtudiant, nom, note, nomMatiere
FROM
    ETUDIANT E,
    MATIERE M,
    NOTE N
WHERE
    N.idEtudiant = E.idEtudiant
        AND M.idMatiere = N.idMatiere;
```

160% 39:9

Filter: File:

idEtudiant	nom	note	nomMatiere
1	Samantha	15	Maths
2	Francis	20	Maths
3	Charles	13	Maths
4	Penelope	14	Maths
5	Craig	10	Maths
6	Steve	8	Maths
10	Patricia	5	Maths
1	Samantha	20	Sociologie
10	Patricia	4	Sociologie
11	Patricia	7	Nutrition

Les identifiants, les noms, les notes et les matières respectives des étudiants ayant une note

Variables dans SQL

```
1  SELECT
2      E1.idEtudiant, E1.nom, E2.idEtudiant, E2.nom
3  FROM
4      ETUDIANT E1,
5      ETUDIANT E2
6  WHERE
7      E1.nom = E2.nom
8          AND E1.idEtudiant <> E2.idEtudiant
```

Les paires d'étudiants qui ont le même nom

160% 43:8

Filter: File:

idEtudiant	nom	idEtudiant	nom
11	Patricia	10	Patricia
10	Patricia	11	Patricia

```
1  SELECT
2      E1.idEtudiant, E1.nom, E2.idEtudiant, E2.nom
3  FROM
4      ETUDIANT E1,
5      ETUDIANT E2
6  WHERE
7      E1.nom = E2.nom
8          AND E1.idEtudiant < E2.idEtudiant
```

Les paires d'étudiants qui ont le même nom sans répétition

160% 28:8

Filter: File:

idEtudiant	nom	idEtudiant	nom
10	Patricia	11	Patricia

Les fonctions d'agrégation dans SQL

Fonctions d'agrégation

```
SELECT FONC_AGGR(COL),...
FROM nomTable1, nomTable2,...
WHERE condition
GROUP BY COL
HAVING condition
```

FONC_AGGR(COL) : MIN| MAX | AVG | SUM | COUNT

La fonction COUNT

Le nombre de notes

```
1 • SELECT
2       COUNT(note)
3   FROM
4       NOTE;
```

10:4

Filter: File:

COUNT(note)
8

Le nombre d'étudiants

```
1 • SELECT
2       COUNT(idEtudiant)
3   FROM
4       ETUDIANT;
```

1:1

Filter: File:

COUNT(idEtudiant)
8

La fonction COUNT

Le nombre d'étudiants avec une note > 10

```sql
SELECT
    COUNT(idEtudiant)
FROM
    NOTE
WHERE
    note > 10
;
```

1:1

Filter: File:

COUNT(idEtudiant)
4

Les fonctions MIN & MAX

La note minimum

```
1  SELECT
2      min(note)
3  FROM
4      NOTE
```

9:4

Filter: File:

min(note)
9

La note maximum

```
1  SELECT
2      max(note)
3  FROM
4      NOTE
```

8:2

Filter: File:

max(note)
20

La fonction AVG

La moyenne des notes

```
1 • SELECT
2       AVG(note)
3   FROM
4       NOTE
```

13:2

Filter: File:

AVG(note)
12.0000

La fonction SUM

La somme des notes

```
1 • SELECT
2       sum(note)
3   FROM
4       NOTE
```

9:4

Filter: File:

sum(note)
96

```
1 • SELECT
2       sum(note)/count(idEtudiant), avg(note)
3   FROM
4       NOTE
```

42:2

Filter: File:

sum(note)/cou...	avg(note)
12.0000	12.0000

La clause GROUP BY

Query 1 | etudiant | matiere | note

```sql
SELECT
    idEtudiant, AVG(NOTE)
FROM
    NOTE
GROUP BY idEtudiant;
```

L'identifiant et la moyenne de chaque étudiant

idEtudiant	AVG(NOTE)
1	17.5000
2	20.0000
3	13.0000
4	14.0000
5	10.0000
6	8.0000
10	4.5000
11	7.0000

La clause GROUP BY

Query 1 | etudiant | matiere | note

```sql
SELECT
    ETUDIANT.idEtudiant, nom, AVG(NOTE)
FROM
    NOTE, ETUDIANT
WHERE NOTE.idEtudiant = ETUDIANT.idEtudiant
GROUP BY ETUDIANT.idEtudiant;
```

L'identifiant, le nom et la moyenne de chaque étudiant

Filter: File:

idEtudiant	nom	AVG(NOTE)
1	Samantha	17.5000
2	Francis	20.0000
3	Charles	13.0000
4	Penelope	14.0000
5	Craig	10.0000
6	Steve	8.0000
10	Patricia	4.5000
11	Patricia	7.0000

La clause HAVING

Query 1 | etudiant | matiere | note

```sql
SELECT
    NOTE.idEtudiant, nom, AVG(note)
FROM
    NOTE,
    ETUDIANT
WHERE
    NOTE.idEtudiant = ETUDIANT.idEtudiant
GROUP BY idEtudiant
HAVING AVG(note) > 10;
```

23:9

Filter: File:

idEtudiant	nom	AVG(note)
1	Samantha	17.5
2	Francis	20
3	Charles	13
4	Penelope	14

L'identifiant, le nom et la moyenne de chaque ayant une moyenne > 10

Opérateurs ensemblistes

UNION

```
1 • SELECT nom as chaine FROM ETUDIANT
2   UNION
3   SELECT nomMatiere as chaine FROM MATIERE;
```

160% 42:3

Filter: File:

chaine
Samantha
Francis
Charles
Penelope
Craig
Steve
Dave
Cameron
Patricia
Maths
Sociologie
Nutrition
Art Moderne
Art Contempo…
Art Classique

Tous les noms des étudiants
et des matières

INTERSECT, EXCEPT

- Clauses non supportées par MySQL
- INTERSECT pour l'intersection de deux ensembles
- EXCEPT pour la différence entre deux ensembles