



# Bases de données

Université Aboubakr Belkaïd de Tlemcen

L3 Génie Industriel

**Questions sur les  
premières parties?**

# Chapitre 4

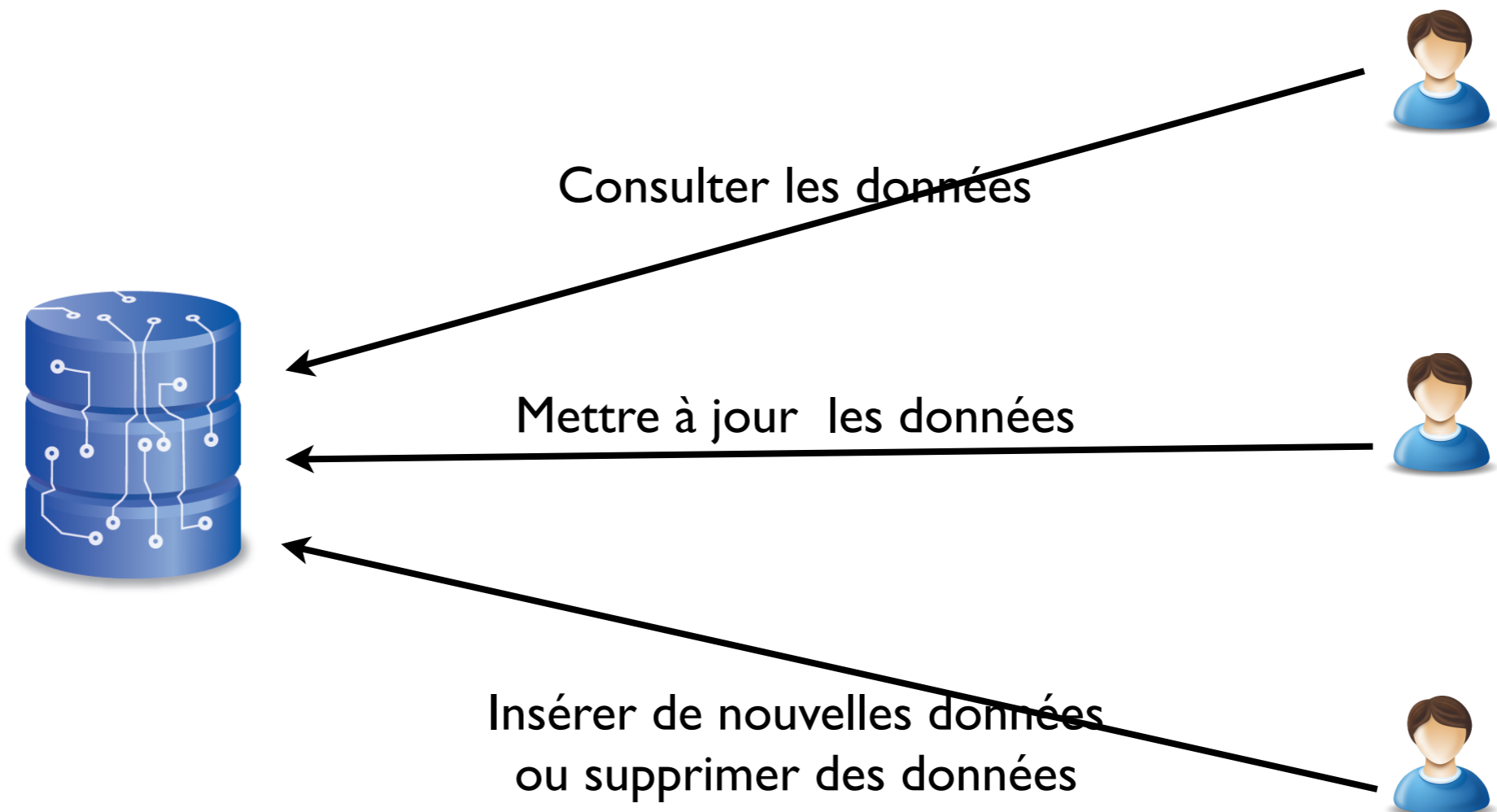
## SQL

# PLAN

- Création de tables
- Insertion de données
- Mise à jour de données
- Suppression de données

# Requête...c'est quoi ?

\* Définition : une requête est une interrogation de la base de données.



# Langages de requêtes

- **Datalog** : pour les bases de données déductives
- **DMX** : pour les modèles d'exploration de données (Data Mining)
- **MDX** : pour les bases de données multidimensionnelles -
- OQL** : pour les bases de données orientées objet
- **Prolog** : pour la programmation logique
- **SPARQL** : pour les graphes RDF
- **XQuery** : pour les données XML
- **XPath** : pour parcourir le DOM
- **SQL** : pour les bases de données relationnelles

# SQL

- Structured Query Language
- Langage de manipulation et de consultation d'une BD
- Supporté par la plupart des SGBD

# Requêtes de création et de manipulation



# Les clauses CREATE/DROP

**CREATE SCHEMA** nomSchema;

**CREATE TABLE** nomTable(  
attribut1 type, attribut2 type, ...  
contrainte1, contrainte2, ...);

type: **INT, TEXT, CHAR, VARCHAR(n),..**

**DROP TABLE** nomTable;

# Contrainte d'intégrité: La clé primaire

**CONSTRAINT** nomContrainte

**PRIMARY KEY** (attribut1 [, attribut2,...] )

# Contrainte d'intégrité: La clé étrangère

**CONSTRAINT** nomContrainte

**FOREIGN KEY** (attribut)

**REFERENCES** nomTable(attribut)

# La clause CREATE

```
Query 1 x note - Table x
1 CREATE TABLE etudiant (
2   idEtudiant INT NOT NULL,
3   nom TEXT NOT NULL,
4   CONSTRAINT pk1 PRIMARY KEY (idEtudiant)
5 )
```

etudiant - Table x

Name: etudiant Schema:

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idEtudiant	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
nom	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column details "

Column Name:  Data

Collation: Table Default De

Comments:

# La clause CREATE

```
Query 1 x etudiant - Table x
1 CREATE TABLE matiere (
2     idMatiere INT NOT NULL,
3     nomMatiere TEXT NOT NULL,
4     CONSTRAINT pk1 PRIMARY KEY (idMatiere)
5 )
```

SCHEMAS

Search objects

- test
- universite
  - Tables
    - etudiant
    - matiere
      - Columns
        - idMatiere
        - nomMatiere
      - Indexes
      - Foreign Keys
      - Triggers
    - Views
    - Routines

Query 1 x matiere - Table x

Name: matiere Schema:

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idMatiere	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
nomMatiere	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column details "

Column Name:  Data

Collation: Table Default De

Comments:

# La clause CREATE

```
1 • CREATE TABLE note (  
2     idEtudiant INT NOT NULL,  
3     idMatiere INT NOT NULL,  
4     note TEXT NOT NULL,  
5     CONSTRAINT pk1 PRIMARY KEY (idEtudiant , idMatiere),  
6     CONSTRAINT fk1 FOREIGN KEY (idEtudiant)  
7         REFERENCES etudiant (idEtudiant),  
8     CONSTRAINT fk2 FOREIGN KEY (idMatiere)  
9         REFERENCES matiere (idMatiere)  
10 )
```

SCHEMAS

Search objects

- test
- universite
  - Tables
    - etudiant
    - matiere
    - note
      - Columns
        - idEtudiant
        - idMatiere
        - note
      - Indexes
      - Foreign Keys
        - fk1
        - fk2
      - Triggers

Query 1 note - Table

Name: note

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idEtudiant	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
idMatiere	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
note	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

## Column details 'idEtudiant'

Column Name:

Collation:

Comments:

# La clause INSERT

**INSERT INTO** nomTable [(col1, col2, ...)]

**VALUES** (valeurAttr1, valeurAttr2, ...)

# La clause INSERT

```
Query 1 x note - Table x
1 • INSERT INTO ETUDIANT VALUES (1, 'Samantha');
2 • INSERT INTO ETUDIANT VALUES (2, 'Francis');
3 • INSERT INTO ETUDIANT VALUES (3, 'Charles');
4 • INSERT INTO ETUDIANT VALUES (4, 'Penelope');
5 • INSERT INTO ETUDIANT VALUES (5, 'Craig');
6 • INSERT INTO ETUDIANT VALUES (6, 'Steve');
7 • INSERT INTO ETUDIANT VALUES (7, 'Dave');
```

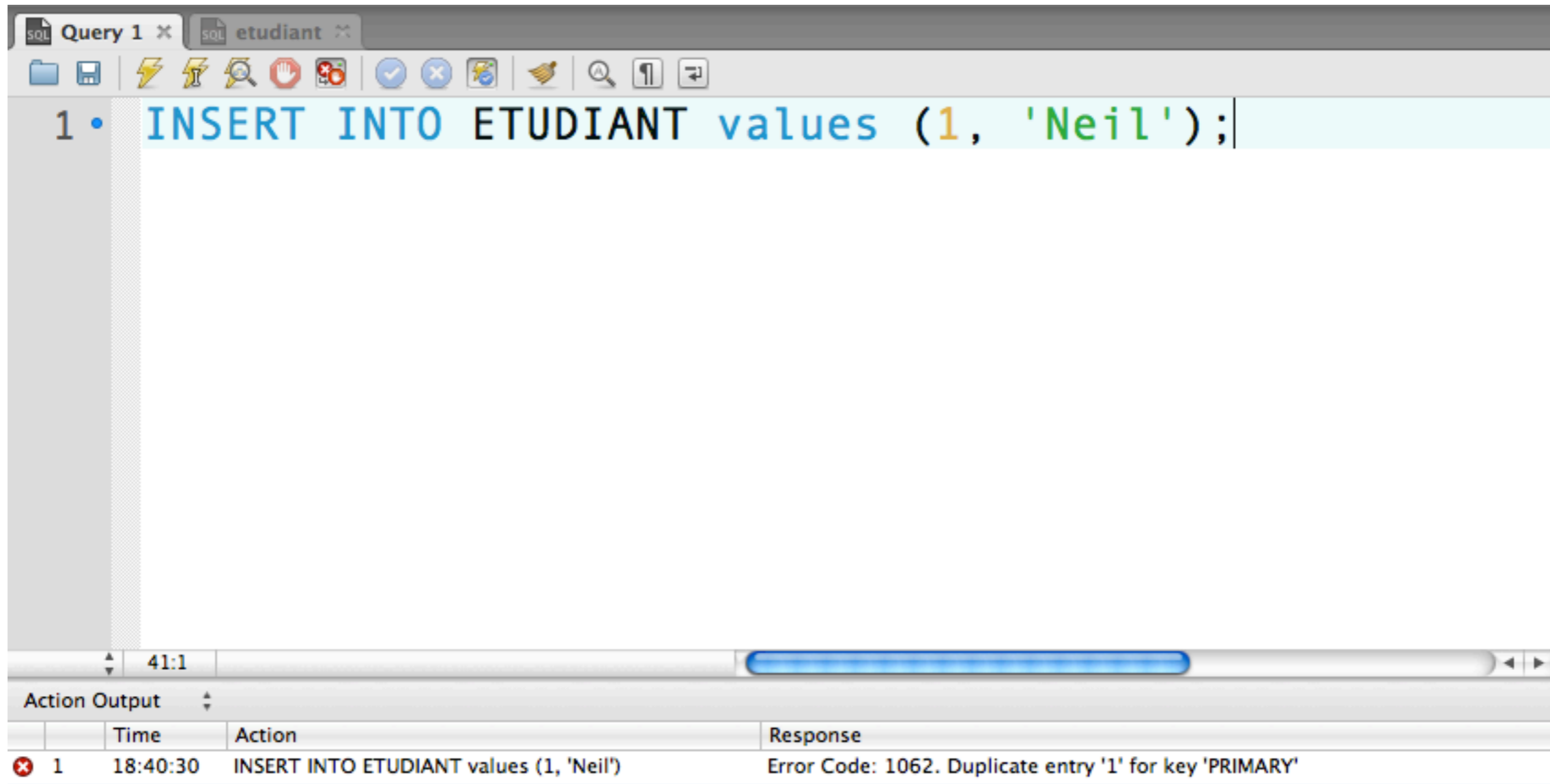
Query 1 x etudiant x

Filter:  Edit: File:

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
	NULL



# La clause INSERT



The screenshot shows a SQL IDE window with two tabs: "Query 1" and "etudiant". The main editor area contains the SQL statement: `INSERT INTO ETUDIANT values (1, 'Neil');`. Below the editor is a scroll bar and an "Action Output" panel. The "Action Output" panel displays a table with the following data:

	Time	Action	Response
❌ 1	18:40:30	INSERT INTO ETUDIANT values (1, 'Neil')	Error Code: 1062. Duplicate entry '1' for key 'PRIMARY'

**Contrainte d'intégrité (clé primaire) violée!**

# La clause INSERT

```
Query 1 x
1 • INSERT INTO MATIERE values (1, 'Maths');
2 • INSERT INTO MATIERE values (2, 'Psychologie');
3 • INSERT INTO MATIERE values (3, 'Nutrition');
4 • INSERT INTO MATIERE values (4, 'Art Moderne');
5 • INSERT INTO MATIERE values (5, 'Art Contemporain');
6 • INSERT INTO MATIERE values (6, 'Art Classique');
7
```

Query 1 x | matiere x

Filter:  Edit: File:

idMatiere	nomMatiere
1	Maths
2	Psychologie
3	Nutrition
4	Art Moderne
5	Art Contempo...
6	Art Classique
	NULL

# La clause UPDATE

**UPDATE** nomTable

**SET** attribut = valeur

**WHERE** condition

# La clause UPDATE

idMatiere	nomMatiere
1	Maths
2	Psychologie
3	Nutrition
4	Art Moderne
5	Art Contempo...
6	Art Classique
	NULL

Avant la requête

```
1 • UPDATE MATIERE
2 SET
3     nomMatiere = 'Sociologie'
4 WHERE
5     idMatiere = 2
```

idMatiere	nomMatiere
1	Maths
2	Sociologie
3	Nutrition
4	Art Moderne
5	Art Contempo...
6	Art Classique
	NULL

Après la requête

# La clause DELETE

**DELETE FROM** nomTable

**WHERE** condition

# La clause DELETE

Query 1 x SQL matiere x

Filter:  Edit: File:

idMatiere	nomMatiere
1	Maths
2	Sociologie
3	Nutrition
4	Art Moderne
5	Art Contempo...
6	Art Classique
	NULL

Avant la requête

Query 1 x SQL matiere x

```
1 • DELETE FROM MATIERE
2 WHERE
3   idMatiere = 2
```

Query 1 x SQL matiere x

Filter:  Edit: File:

idMatiere	nomMatiere
1	Maths
3	Nutrition
4	Art Moderne
5	Art Contempo...
6	Art Classique
	NULL

Après la requête

# Requêtes de consultation

# La clause SELECT

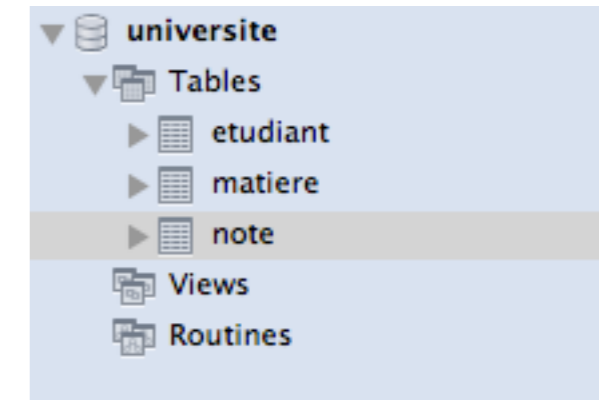
**SELECT**      attribut1, attribut2, ...

**FROM**        table1, table2, ...

**WHERE**        condition



# La clause SELECT



idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
	NULL

idMatiere	nomMatiere
1	Maths
2	Sociologie
3	Nutrition
4	Art moderne
5	Art contempo...
6	Art classique
	NULL

idEtudiant	idMatiere	note
1	1	15
1	2	20
2	1	20
3	1	13
4	1	14
5	1	10
6	1	8
	NULL	NULL



# La clause SELECT - Opérateur de sélection

```
1 • SELECT
2     *
3 FROM
4     ETUDIANT
5 WHERE
6     idEtudiant > 4;
7
```

Les étudiants avec un identifiant > 4

$\sigma_{idEtudiant > 4} ETUDIANT$

1:1

Filter:  Edit: File:

idEtudiant	nom
5	Craig
6	Steve
7	Dave
	NULL

# La clause SELECT - Opérateur de sélection

Les étudiants qui s'appellent Samantha

$\sigma_{nom = 'Samantha'} ETUDIANT$

```
1 • SELECT
2     *
3 FROM
4     ETUDIANT
5 WHERE
6     nom = 'Samantha';
7
```

8:6

Filter:  Edit: File:

idEtudiant	nom
▶ 1	Samantha
	NULL

# La clause SELECT - Opérateur de sélection

Les étudiants dont le nom commence avec 'S'

$\sigma_{nom \text{ like } 'S\%'} ETUDIANT$

```
1 • SELECT
2     *
3 FROM
4     ETUDIANT
5 WHERE
6     nom like 'S%';
7
```

19:6

Filter:  Edit: File:

idEtudiant	nom
▶ 1	Samantha
6	Steve
	NULL

# La clause SELECT - Opérateur de projection

The image shows a screenshot of a SQL query editor window titled "Query 1". The editor contains the following SQL code:

```
1 • SELECT
2     nom
3 FROM
4     ETUDIANT
```

Handwritten blue text next to the code reads: "Les noms des étudiants" and the mathematical expression  $\pi_{nom}ETUDIANT$ .

Below the editor, the result set is displayed in a table with a single column named "nom". The results are:

nom
Samantha
Francis
Charles
Penelope
Craig
Steve
Dave







# La clause SELECT - Opérateur de projection

The screenshot shows a SQL query editor window titled "Query 1". The query text is:

```
1 • SELECT
2     note/2
3 FROM
4     NOTE
```

Handwritten blue text next to the query reads: "Les notes divisées par 2".

Below the query editor, the results are displayed in a table with one column labeled "note/2". The results are:

note/2
10.0000
6.5000
7.0000
5.0000
4.0000
4.0000

# La clause SELECT - Opérateur de projection

## Notion d'alias

The screenshot shows a SQL query editor window titled "Query 1". The query text is as follows:

```
1 • SELECT
2     note/2 as noteSur10
3 FROM
4     NOTE
```

A handwritten note in blue ink, "Les notes divisées par 2", is written next to the `note/2` expression in the query. Below the query editor, the results of the query are displayed in a table with one column named "noteSur10". The results are:

noteSur10
10.0000
6.5000
7.0000
5.0000
4.0000
4.0000

# Le produit cartésien et les jointures

The screenshot shows a SQL query editor window titled "Query 1". The query text is:

```
1 • SELECT
2     *
3 FROM
4     ETUDIANT, NOTE
```

Handwritten in blue on the right side of the editor is "ETUDIANT X NOTE".

Below the query editor, the result set is displayed in a table with 5 columns: idEtudiant, nom, idEtudiant, idMatiere, and note. The table contains 15 rows, representing the Cartesian product of the ETUDIANT and NOTE tables.

idEtudiant	nom	idEtudiant	idMatiere	note
1	Samantha	2	1	20
1	Samantha	3	1	13
1	Samantha	4	1	14
1	Samantha	5	1	10
1	Samantha	6	1	8
1	Samantha	1	4	8
2	Francis	2	1	20
2	Francis	3	1	13
2	Francis	4	1	14
2	Francis	5	1	10
2	Francis	6	1	8
2	Francis	1	4	8
3	Charles	2	1	20
3	Charles	3	1	13
3	Charles	4	1	14
3	Charles	5	1	10

# Le produit cartésien et les jointures

```
1 • SELECT
2     nom
3 FROM
4     ETUDIANT,
5     NOTE
6 WHERE
7     ETUDIANT.idEtudiant = NOTE.idEtudiant
8
```

Le nom de tous les étudiants  
ayant une note

$\pi_{nom} ETUDIANT \bowtie NOTE$

The screenshot shows the result of the SQL query in a database interface. The main window displays a list of names: Samantha, Francis, Charles, Penelope, Craig, and Steve. Below this, two smaller windows show the source tables. The 'etudiant' table lists 7 students with their IDs and names. The 'note' table lists 6 notes with student IDs, subject IDs, and scores.

nom
Samantha
Samantha
Francis
Charles
Penelope
Craig
Steve

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
	NULL

idEtudiant	idMatiere	note
1	1	15
1	2	20
2	1	20
3	1	13
4	1	14
5	1	10
6	1	8
	NULL	NULL



# Le produit cartésien et les jointures

```
SQL Query 1 x
SELECT distinct
  nom
FROM
  ETUDIANT,
  NOTE
WHERE
  ETUDIANT.idEtudiant = NOTE.idEtudiant
  AND note > 15
```

Le nom de tous les étudiants ayant une note > 15 sans duplication

$\pi_{nom}(\sigma_{note > 15} ETUDIANT \bowtie NOTE)$

nom
Samantha
Francis

idEtudiant	nom
1	Samantha
2	Francis
3	Charles
4	Penelope
5	Craig
6	Steve
7	Dave
	NULL

idEtudiant	idMatiere	note
1	1	15
1	2	20
2	1	20
3	1	13
4	1	14
5	1	10
6	1	8
	NULL	NULL



# Le produit cartésien et les jointures

```
Query 1 x
SELECT
  nom, note, nomMatiere
FROM
  ETUDIANT,
  NOTE,
  MATIERE
WHERE
  ETUDIANT.idEtudiant = NOTE.idEtudiant
  AND MATIERE.idMatiere = NOTE.idMatiere
  AND note > 15
```

Query 1 x

Filter:  File:

nom	note	nomMatiere
Francis	20	Maths
Samantha	20	Sociologie

Le nom, la note et la matière de tous les étudiants ayant une note > 15

$\pi_{nom,note,nomMatiere}(\sigma_{note>15}ETUDIANT \bowtie NOTE \bowtie MATIERE)$



# Le produit cartésien et les jointures

```
SQL Query 1 x
SELECT
  nom, note
FROM
  ETUDIANT,
  NOTE,
  MATIERE
WHERE
  ETUDIANT.idEtudiant = NOTE.idEtudiant
  AND MATIERE.idMatiere = NOTE.idMatiere
  AND note > 10
  AND nomMatiere = 'Maths'
```

Le nom et la note de tous les étudiants ayant une note >10 en Maths

nom	note
Samantha	15
Francis	20
Charles	13
Penelope	14

$\pi_{nom,note,nomMatiere}$   
 $(\sigma_{note>10 \wedge nomMatiere='Maths'} ETUDIANT \bowtie NOTE \bowtie MATIERE)$

# Le produit cartésien et les jointures

```
1 • SELECT
2     idEtudiant, nom, note
3 FROM
4     ETUDIANT,
5     NOTE,
6     MATIERE
7 WHERE
8     ETUDIANT.idEtudiant = NOTE.idEtudiant
9     AND MATIERE.idMatiere = NOTE.idMatiere
10    AND note > 10
11    AND nomMatiere = 'Maths'
```

L'identifiant, le nom et la note de  
tous les étudiants ayant une note >10  
en Maths

160% 19:8

Action Output			
	Time	Action	Response
✖ 1	14:02:03	SELECT idEtudiant, nom, note FROM ETUDIANT, NOTE, ...	Error Code: 1052. Column 'idEtudiant' in field list is ambiguous

**La colonne idEtudiant est ambiguë!**



# Le produit cartésien et les jointures

```
Query 1 x
SELECT
  ETUDIANT.idEtudiant, nom, note
FROM
  ETUDIANT,
  NOTE,
  MATIERE
WHERE
  ETUDIANT.idEtudiant = NOTE.idEtudiant
  AND MATIERE.idMatiere = NOTE.idMatiere
  AND note > 10
  AND nomMatiere = 'Maths'
ORDER BY note;
```

Query 1 x

Filter:  File:

idEtudiant	nom	note
3	Charles	13
4	Penelope	14
1	Samantha	15
2	Francis	20

L'identifiant, le nom et la note de tous les étudiants ayant une note >10 en Maths, par ordre ascendant des notes

# Le produit cartésien et les jointures

```
Query 1 x
SELECT
  ETUDIANT.idEtudiant, nom, note
FROM
  ETUDIANT,
  NOTE,
  MATIERE
WHERE
  ETUDIANT.idEtudiant = NOTE.idEtudiant
  AND MATIERE.idMatiere = NOTE.idMatiere
  AND note > 10
  AND nomMatiere = 'Maths'
ORDER BY note desc;
```

Query 1 x

Filter:  File:

idEtudiant	nom	note
2	Francis	20
1	Samantha	15
4	Penelope	14
3	Charles	13

L'identifiant, le nom et la note de tous les étudiants ayant une note >10 en Maths, par ordre descendant des notes