Département d'Informatique
Cours : Système de gestion de bases de données
Semestre 5
Encadrant : Denis Lubin
L3_Informatique_Miage
2025-2026

ANALYSE MÉTHODE ENTITÉ-RELATION

POUR LA GESTION D'UNE PLATEFORME UNIVERSITAIRE

MEMBRE DU GROUPE :

TAHAR BELLAR AICHA 20231526

EL HATHOUT LINA 20231126

SOMMAIRE

- INTRODUCTION

Ce rapport se concentre sur la modélisation d'une plateforme universitaire destinée à gérer les principaux acteurs et objets d'un cursus : étudiants, enseignants, cours, examens, inscriptions et notes.

L'objectif est de concevoir un Modèle Conceptuel de Données (MCD) selon la méthode Entité-Relation, qui constituera le socle d'une base de données relationnelle fiable et structurée.

Le rapport proposera également diverses requêtes SQL illustrant l'exploitation de la base, aussi bien pour des extractions courantes que pour des analyses avancées portant sur les inscriptions, les résultats et la distribution des étudiants.

- IDENTIFICATION DES ENTITÉS PRINCIPALES :

ENTITÉ	ATTRIBUTS PRINCIPAUX
Etudiant	id_etudiant (PK), prenom, date_naissance, email, nom
Enseignant	id_enseignant (PK), nom, prenom, email
Cours	id_cours (PK), nom_cours, id_enseignant(FK)
Examen	id_examen (PK), nom_examen, date_examen, semestre, id_cours(FK), id_enseignant(FK)
Inscription	id_inscription (PK), date_inscription, annulée, id_etudiant(FK), id_cours(FK)
Note	id_note (PK), valeur, id_etudiant(FK), id_examen(FK)

- DÉFINITION DES RELATIONS ENTRE LES ENTITÉS :

  - Cours-enseignant :

Un cours est dispensé par un seul enseignant ; un enseignant peut dispenser plusieurs cours.

  - Cours-examen :

Un cours peut avoir plusieurs examens ; un examen appartient à un seul cours.

  - Enseignant-examen :

Un enseignant peut encadrer plusieurs examens ; un examen est encadré par un seul enseignant.

  - Étudiant-cours :

Un étudiant peut s'inscrire à plusieurs cours ; un cours peut avoir plusieurs étudiants inscrits.

  - Étudiant-examen-note :

Un étudiant peut avoir plusieurs notes ; un examen peut générer plusieurs notes (car plusieurs étudiants passent le même examen).

- DÉTAIL DES CARDINALITÉS DES RELATIONS:

  - **Cours-enseignant :**

Un cours est dispensé par un seul enseignant (1,1)
Un enseignant peut dispenser plusieurs cours (0,N)

  - **Cours-examen :**

Un cours peut avoir plusieurs examens (1,N)
Un examen appartient à un seul cours (1,1)

  - **Enseignant-examen :**

Un enseignant peut encadrer plusieurs examens (0,N)
Un examen est encadré par un seul enseignant (1,1)

- **Étudiant-cours :**

Un étudiant peut s'inscrire à plusieurs cours (1,N)
Un cours peut avoir plusieurs étudiants inscrits (1,N)

- **Étudiant-examen-note :**

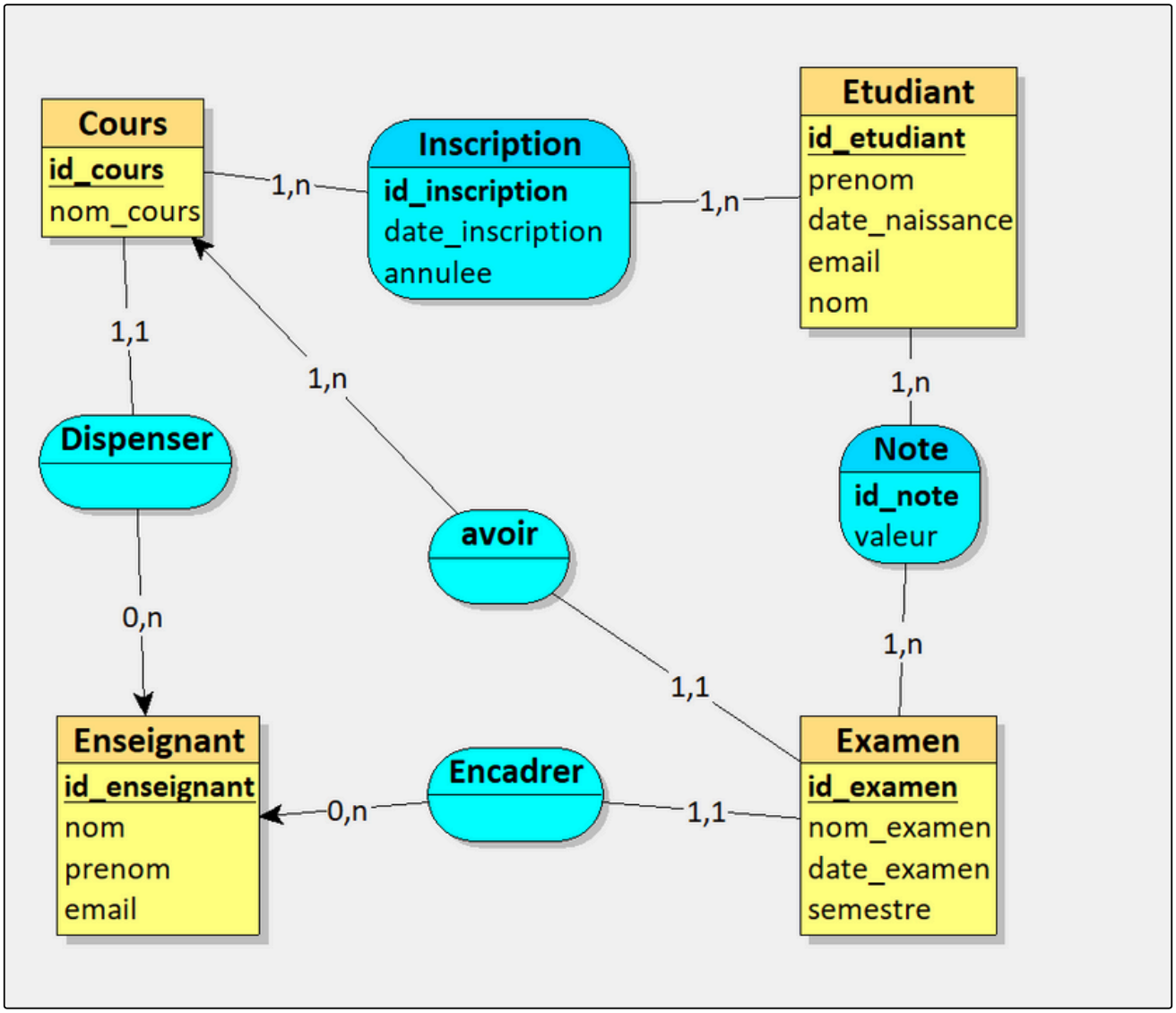
Un étudiant peut avoir plusieurs notes (1,N)
Un examen peut avoir plusieurs notes (1,N)

- IDENTIFICATION DES CLÉS PRIMAIRES ET ÉTRANGÈRES :

ENTITÉ	CLÉ PRIMAIRE (PK)	CLÉ(S) ÉTRANGÈRE(S) (FK)
Etudiant	id_etudiant	_
Enseignant	id_enseignant	_
Cours	id_cours	id_enseignant
Examen	id_examen	id_cours, id_enseignant
Inscription	Id_inscription	id_etudiant, id_cours
Note	id_note	id_etudiant, id_examen

- DIAGRAMME MCD DE LA PLATEFORME DE GESTION UNIVERSITAIRE

Le modèle conceptuel de données (MCD) ci-dessus structure la gestion universitaire autour de six entités (dont 2 associatives) et de plusieurs relations spécifiques :



Entités principales

- Etudiant, Enseignant, Cours, Examen, Inscription, Note: Toutes les informations nécessaires à la gestion académique sont réparties dans ces entités.

Clés primaires et étrangères

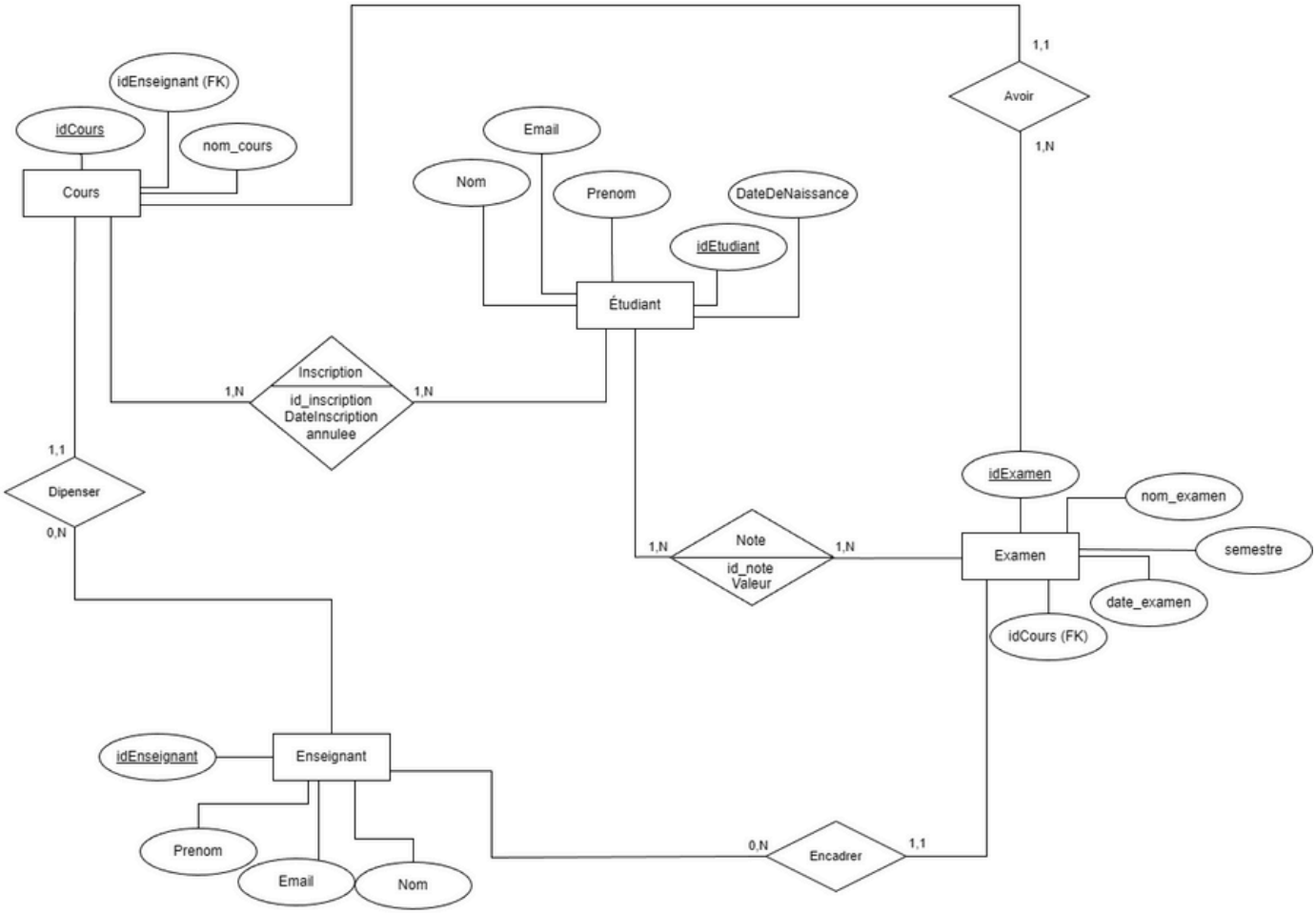
- L'utilisation de clés primaires sur chaque entité garantit la non-duplication des informations.

- Les clés étrangères assurent la cohérence des relations, facilitant l'intégrité lors des opérations sur la base relationnelle.

Relations

- Examen est rattaché à la fois au cours (pour indiquer sa discipline) et à l'enseignant (qui l'encadre).
- Inscription fait le lien entre l'étudiant et le cours afin de modéliser l'association dynamique.
- Note relie les étudiants aux examens : un étudiant reçoit plusieurs notes pour différents examens.

- MODÈLE ENTITÉ-RELATION DE LA PLATEFORME UNIVERSITAIRE



Création des tables :

```
CREATE DATABASE projetSGBD;

CREATE TABLE Enseignant (
   id_enseignant INT PRIMARY KEY AUTO_INCREMENT,
   nom VARCHAR(50) NOT NULL,
   prenom VARCHAR(50) NOT NULL,
   email VARCHAR(50)
);


CREATE TABLE Etudiant (
   id_etudiant INT PRIMARY KEY AUTO_INCREMENT,
   nom VARCHAR(50) NOT NULL,
   prenom VARCHAR(50) NOT NULL,
   date_naissance DATE NOT NULL,
   email VARCHAR(50)
)  ;


CREATE TABLE Cours (
   id_cours INT PRIMARY KEY AUTO_INCREMENT,
   nom_cours VARCHAR(50) NOT NULL,
   id_enseignant INT NOT NULL,
                  FOREIGN KEY (id_enseignant) REFERENCES
Enseignant(id_enseignant)
      ON DELETE RESTRICT ON UPDATE CASCADE
);
```

```
CREATE TABLE Examen (
    id_examen INT PRIMARY KEY AUTO_INCREMENT,
    nom_examen VARCHAR(50) NOT NULL,
    date_examen DATE NOT NULL,
    id_cours INT NOT NULL,
    id_enseignant INT NOT NULL,
    semestre INT,
    FOREIGN KEY (id_cours) REFERENCES Cours(id_cours)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (id_enseignant) REFERENCES Enseignant(id_enseignant)
        ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE Inscription (
    id_inscription INT PRIMARY KEY AUTO_INCREMENT,
    id_etudiant INT NOT NULL,
    id_cours INT NOT NULL,
    date_inscription DATE NOT NULL,
    annulee BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (id_etudiant) REFERENCES Etudiant(id_etudiant)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (id_cours) REFERENCES Cours(id_cours)
        ON DELETE CASCADE ON UPDATE CASCADE,
     -- Contrainte d'unicité : un étudiant ne peut s'inscrire qu'une fois à un
cours
    UNIQUE KEY unique_etudiant_cours (id_etudiant, id_cours)
);
CREATE TABLE Note (
    id_note INT PRIMARY KEY AUTO_INCREMENT,
    id_etudiant INT,
    id_examen INT,
    valeur DECIMAL(4,2),
    FOREIGN KEY (id_etudiant) REFERENCES Etudiant(id_etudiant)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (id_examen) REFERENCES Examen(id_examen)
        ON DELETE CASCADE ON UPDATE CASCADE,
       -- Contrainte d'unicité : un étudiant ne peut avoir qu'une note par
examen
    UNIQUE KEY unique_etudiant_examen (id_etudiant, id_examen)
);
```

Remplissage des tables :

```
INSERT INTO Enseignant (nom, prenom, email) VALUES
('Dupont', 'Jean', 'jean.dupont@email.com'),
('Martin', 'Claire', 'claire.martin@email.com'),
('Lemoine', 'Pierre', 'pierre.lemoine@email.com'),
('Tanguy', 'Lucie', 'lucie.tanguy@email.com'),
('Leclerc', 'Michel', 'michel.leclerc@email.com'),
('Bernard', 'Sophie', 'sophie.bernard@email.com'),
('Garnier', 'Antoine', 'antoine.garnier@email.com'),
('Brun', 'Elodie', 'elodie.brun@email.com'),
('Richard', 'Lucas', 'lucas.richard@email.com'),
('Fournier', 'Nathalie', 'nathalie.fournier@email.com');

INSERT INTO Etudiant (nom, prenom, date_naissance, email) VALUES
('Durand', 'Alice', '1998-04-15', 'alice.durand@email.com'),
('Lefevre', 'Benoit', '1997-10-22', 'benoit.lefevre@email.com'),
('Gauthier', 'Sophie', '1999-07-30', 'sophie.gauthier@email.com'),
('Petit', 'Marc', '1996-01-10', 'marc.petit@email.com'),
('Sanchez', 'Carla', '1998-11-25', 'carla.sanchez@email.com'),
('Lemoine', 'Julien', '2000-02-18', 'julien.lemoine@email.com'),
('Rossi', 'Giovanni', '1997-04-05', 'giovanni.rossi@email.com'),
('Girard', 'Nicolas', '1998-09-08', 'nicolas.girard@email.com'),
('Blanc', 'Hélène', '2000-12-14', 'helene.blanc@email.com'),
('David', 'Paul', '2007-03-23', 'paul.david@email.com');

INSERT INTO Cours (nom_cours, id_enseignant) VALUES
('Mathématiques', 1),  -- Enseigné par Jean Dupont
('Informatique', 2),   -- Enseigné par Claire Martin
('Physique', 3),       -- Enseigné par Pierre Lemoine
('Chimie', 4),         -- Enseigné par Lucie Tanguy
('Biologie', 5),       -- Enseigné par Michel Leclerc
('Économie', 6),       -- Enseigné par Sophie Bernard
('Histoire', 7),       -- Enseigné par Antoine Garnier
('Géographie', 8),     -- Enseigné par Elodie Brun
('Philosophie', 9),    -- Enseigné par Lucas Richard
('Anglais', 10),       -- Enseigné par Nathalie Fournier
('Mathématiques2', 1);  -- Enseigné par Jean Dupont
```

```
INSERT INTO Examen (nom_examen, date_examen, id_cours, id_enseignant, semestre) VALUES
('Examen de Mathématiques 1', '2025-10-10', 1, 1, 1),
('Examen Informatique 1', '2025-06-05', 2, 2, 1),
('Examen de Physique 1', '2025-07-15', 3, 3, 1),
('Examen de Chimie 1', '2025-06-15', 4, 4, 2),
('Examen de Biologie 1', '2025-07-20', 5, 5, 1),
('Examen Économie 1', '2025-05-20', 6, 6, 2),
('Examen Histoire 1', '2025-08-01', 7, 7, 1),
('Examen de Géographie 1', '2025-06-25', 8, 8, 1),
('Examen de Philosophie 1', '2025-07-10', 9, 9, 1),
('Examen Anglais 1', '2025-05-30', 10, 10, 1),
('Examen de Mathématiques 2', '2026-03-10', 11, 1, 2);

INSERT INTO Inscription (id_etudiant, id_cours, date_inscription) VALUES
(1, 1, '2025-09-01'),  -- Alice Durand s'inscrit à Mathématiques
(2, 2, '2025-03-12'),  -- Benoit Lefevre s'inscrit à Informatique
(3, 3, '2025-01-25'),  -- Sophie Gauthier s'inscrit à Physique
(4, 4, '2025-03-15'),  -- Marc Petit s'inscrit à Chimie
(5, 5, '2025-02-20'),  -- Carla Sanchez s'inscrit à Biologie
(6, 6, '2025-01-30'),  -- Julien Lemoine s'inscrit à Économie
(7, 7, '2025-04-01'),  -- Giovanni Rossi s'inscrit à Histoire
(8, 8, '2025-02-18'),  -- Nicolas Girard s'inscrit à Géographie
(9, 9, '2025-03-28'),  -- Hélène Blanc s'inscrit à Philosophie
(10, 10, '2025-05-05'), -- Paul David s'inscrit à Anglais
(1, 11, '2025-01-10');  -- Alice Durand s'inscrit à Mathématiques2

INSERT INTO Note (id_etudiant, id_examen, valeur) VALUES
(1, 1, 15.50),  -- Alice Durand obtient 15.50 à l'examen de Mathématiques 1
(2, 2, 12.00),  -- Benoit Lefevre obtient 12.00 à l'examen d'Informatique
(3, 3, 18.00),  -- Sophie Gauthier obtient 18.00 à l'examen de Physique
(4, 4, 7.00),   -- Marc Petit obtient 7.00 à l'examen de Chimie
(5, 5, 17.50),  -- Carla Sanchez obtient 17.50 à l'examen de Biologie
(6, 6, 6.00),   -- Julien Lemoine obtient 6.00 à l'examen d'Économie
(7, 7, 16.00),  -- Giovanni Rossi obtient 16.00 à l'examen d'Histoire
(9, 9, 18.50),  -- Hélène Blanc obtient 18.50 à l'examen de Philosophie
(1, 10, 9.00),  -- Paul David obtient 9.00 à l'examen d'Anglais
(1, 11, 17);  -- Alice Durand obtient 17 à l'examen de Mathématiques 2
```

LES REQUETES :

-- Requête 1 :

```
SELECT DISTINCT e.id_etudiant, e.nom, e.prenom
FROM Etudiant e
LEFT JOIN Note n ON e.id_etudiant = n.id_etudiant
WHERE n.id_note IS NULL;
```

-- Requête 2 :

```
SELECT COUNT(*) AS nombre_total_etudiants
FROM Etudiant;
```

-- Requête 3 :

```
SELECT e.id_etudiant, e.nom, e.prenom, c.nom_cours,
      m1.moyenne AS moyenne_semestre_precedent,
      m2.moyenne AS moyenne_semestre_suivant
FROM Etudiant e
JOIN Cours c
-- Moyenne du semestre précédent
JOIN (
       SELECT n.id_etudiant, ex.id_cours, ex.semestre, AVG(n.valeur) AS
moyenne
   FROM Note n
   JOIN Examen ex ON n.id_examen = ex.id_examen
   GROUP BY n.id_etudiant, ex.id_cours, ex.semestre
) m1 ON e.id_etudiant = m1.id_etudiant AND c.id_cours = m1.id_cours
-- Moyenne du semestre suivant
JOIN (
       SELECT n.id_etudiant, ex.id_cours, ex.semestre, AVG(n.valeur) AS
moyenne
   FROM Note n
   JOIN Examen ex ON n.id_examen = ex.id_examen
   GROUP BY n.id_etudiant, ex.id_cours, ex.semestre
) m2 ON e.id_etudiant = m2.id_etudiant
     AND c.id_cours = m2.id_cours
     AND m2.semestre = m1.semestre + 1
WHERE m2.moyenne > m1.moyenne;
```

```
-- Requête 4 :
SELECT
    c.id_cours,
    c.nom_cours,
    AVG(n.valeur) AS moyenne_notes
FROM Cours c
JOIN Examen ex ON c.id_cours = ex.id_cours
JOIN Note n ON ex.id_examen = n.id_examen
GROUP BY c.id_cours, c.nom_cours;

-- Requête 5 :
SELECT DISTINCT
    en.id_enseignant,
    en.nom,
    en.prenom,
    ex.nom_examen
FROM Enseignant en
JOIN Examen ex ON en.id_enseignant = ex.id_enseignant
ORDER BY en.nom, ex.nom_examen;

-- Requête 6 :
SELECT
    c.id_cours,
    c.nom_cours,
    COUNT(ex.id_examen) AS nombre_examens
FROM Cours c
LEFT JOIN Examen ex ON c.id_cours = ex.id_cours
GROUP BY c.id_cours, c.nom_cours;

-- Requête 7 :
SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, date_naissance, CURDATE()) < 20 THEN
'Moins de 20 ans'
         WHEN TIMESTAMPDIFF(YEAR, date_naissance, CURDATE()) BETWEEN
20 AND 30 THEN '20-30 ans'
       ELSE 'Plus de 30 ans'
    END AS tranche_age,
    COUNT(*) AS nombre_etudiants
FROM Etudiant
GROUP BY
    tranche_age;
```

Requête 8 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    AVG(n.valeur) AS moyenne_generale
FROM Etudiant e
JOIN Note n ON e.id_etudiant = n.id_etudiant
GROUP BY e.id_etudiant, e.nom, e.prenom
HAVING moyenne_generale > 15;
```

-- Requête 9 :

```
SELECT
    en.id_enseignant,
    en.nom,
    en.prenom
FROM Enseignant en
LEFT JOIN Cours c ON en.id_enseignant = c.id_enseignant
WHERE c.id_cours IS NULL;
```

-- Requête 10 :

```
SELECT
    c.id_cours,
    c.nom_cours
FROM Cours c
JOIN Enseignant en ON c.id_enseignant = en.id_enseignant
WHERE en.nom = 'Dupont';
```

-- Requête 11 :

```
SELECT
    c.id_cours,
    c.nom_cours,
    COUNT(i.id_inscription) AS nombre_inscriptions
FROM Cours c
LEFT JOIN Inscription i ON c.id_cours = i.id_cours AND i.annulee = 0
GROUP BY c.id_cours, c.nom_cours
ORDER BY nombre_inscriptions DESC;
```

Requête 12 :

```
SELECT e.id_etudiant, e.nom, e.prenom, ex.nom_examen, n.valeur AS note
FROM Note n
JOIN Examen ex ON n.id_examen = ex.id_examen
JOIN Etudiant e ON n.id_etudiant = e.id_etudiant
WHERE n.valeur = (
    SELECT MAX(n2.valeur)
    FROM Note n2
    WHERE n2.id_examen = n.id_examen
)
ORDER BY ex.id_examen;
```

-- Requête 13 :

```
SELECT
    i.id_inscription,
    e.nom,
    e.prenom,
    c.nom_cours,
    i.date_inscription
FROM Inscription i
JOIN Etudiant e ON i.id_etudiant = e.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
WHERE i.date_inscription > '2025-01-01';
```

-- Requête 14 :

```
SELECT
    AVG(nombre_inscriptions) AS moyenne_inscriptions_par_etudiant
FROM (
    SELECT
        id_etudiant,
        COUNT(*) AS nombre_inscriptions
    FROM Inscription
    WHERE annulee = 0
    GROUP BY id_etudiant
) AS sous_requete;
```

Requête 15 :

```
SELECT
    i.id_inscription,
    c.nom_cours,
    i.date_inscription
FROM Inscription i
JOIN Etudiant e ON i.id_etudiant = e.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
WHERE e.nom = 'Durand' AND e.prenom = 'Alice';
```

-- Requête 16 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    e.email
FROM Etudiant e
JOIN Inscription i ON e.id_etudiant = i.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
WHERE c.nom_cours = 'Informatique' ;
```

-- Requête 17 :

```
SELECT
    YEAR(date_inscription) AS annee,
    MONTH(date_inscription) AS mois,
    COUNT(*) AS nombre_inscriptions
FROM Inscription
WHERE YEAR(date_inscription) = 2025
GROUP BY YEAR(date_inscription), MONTH(date_inscription)
ORDER BY annee, mois;
```

Requête 18 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    c.nom_cours,
    AVG(n.valeur) AS moyenne_cours
FROM Etudiant e
JOIN Inscription i ON e.id_etudiant = i.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
LEFT JOIN Examen ex ON c.id_cours = ex.id_cours
LEFT JOIN Note n ON ex.id_examen = n.id_examen AND n.id_etudiant =
e.id_etudiant
WHERE i.annulee = 0
GROUP BY e.id_etudiant, e.nom, e.prenom, c.id_cours, c.nom_cours;
```

-- Requête 19 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    COUNT(i.id_inscription) AS nombre_cours
FROM Etudiant e
JOIN Inscription i ON e.id_etudiant = i.id_etudiant
WHERE i.annulee = 0
GROUP BY e.id_etudiant, e.nom, e.prenom
HAVING COUNT(i.id_inscription) > 3;
```

-- Requête 20 :

```
SELECT
    en.id_enseignant,
    en.nom,
    en.prenom,
    c.id_cours,
    c.nom_cours
FROM Enseignant en
LEFT JOIN Cours c ON en.id_enseignant = c.id_enseignant
ORDER BY en.nom, c.nom_cours;
```

Requête 21 :

```
SELECT
    i.id_inscription,
    e.nom,
    e.prenom,
    c.nom_cours,
    i.date_inscription
FROM Inscription i
JOIN Etudiant e ON i.id_etudiant = e.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
WHERE i.annulee = 1;
```

-- Requête 22 :

```
SELECT e.id_enseignant, e.nom, e.prenom, COUNT(c.id_cours) AS nb_cours
FROM Enseignant e
JOIN Cours c ON e.id_enseignant = c.id_enseignant
GROUP BY e.id_enseignant, e.nom, e.prenom
ORDER BY nb_cours DESC
LIMIT 1;
```

-- Requête 23 :

```
SELECT *
FROM Etudiant;
```

-- Requête 24 :

```
SELECT
    c.id_cours,
    c.nom_cours,
    COUNT(i.id_inscription) AS nombre_inscriptions
FROM Cours c
LEFT JOIN Inscription i ON c.id_cours = i.id_cours AND i.annulee = 0
GROUP BY c.id_cours, c.nom_cours;
```

Requête 25 :

```
SELECT
   c.id_cours,
   c.nom_cours,
   COUNT(i.id_inscription) AS nombre_etudiants
FROM Cours c
JOIN Inscription i ON c.id_cours = i.id_cours
WHERE i.annulee = 0
GROUP BY c.id_cours, c.nom_cours
HAVING COUNT(i.id_inscription) > 50;
```

-- Requête 26 :

```
SELECT
   c.id_cours,
   c.nom_cours,
   COUNT(CASE WHEN n.valeur >= 10 THEN 1 END) AS nb_reussite,
   COUNT(i.id_inscription) AS nb_inscriptions,
        ROUND(COUNT(CASE WHEN n.valeur >= 10 THEN 1 END) /
COUNT(i.id_inscription) * 100, 2) AS taux_reussite
FROM Cours c
LEFT JOIN Inscription i ON c.id_cours = i.id_cours AND i.annulee = 0
LEFT JOIN Note n ON i.id_etudiant = n.id_etudiant
LEFT JOIN Examen ex ON n.id_examen = ex.id_examen AND ex.id_cours
= c.id_cours
GROUP BY c.id_cours, c.nom_cours
ORDER BY taux_reussite DESC
LIMIT 1;
```

-- Requête 27 :

```
SELECT
   YEAR(date_inscription) AS annee,
   MONTH(date_inscription) AS mois,
   COUNT(*) AS nombre_annulations
FROM Inscription
WHERE annulee = 1
GROUP BY YEAR(date_inscription), MONTH(date_inscription)
ORDER BY annee, mois;
```

Requête 28 :

```
SELECT e.id_etudiant, e.nom, e.prenom
FROM Etudiant e
JOIN Inscription i ON e.id_etudiant = i.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
WHERE c.id_enseignant = 2
GROUP BY e.id_etudiant, e.nom, e.prenom
HAVING COUNT(DISTINCT c.id_cours) = (
    SELECT COUNT(*)
    FROM Cours
    WHERE id_enseignant = 2
);
```

-- Requête 29 :

```
SELECT
    e.id_etudiant,
    e.nom AS nom_etudiant,
    e.prenom AS prenom_etudiant,
    c.nom_cours,
    en.nom AS nom_enseignant,
    en.prenom AS prenom_enseignant,
    i.date_inscription
FROM Inscription i
JOIN Etudiant e ON i.id_etudiant = e.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
JOIN Enseignant en ON c.id_enseignant = en.id_enseignant
WHERE i.annulee = 0
ORDER BY e.nom, c.nom_cours;
```

-- Requête 30 :

```
SELECT
    c.id_cours,
    c.nom_cours,
    AVG(n.valeur) AS moyenne_notes
FROM Cours c
JOIN Examen ex ON c.id_cours = ex.id_cours
JOIN Note n ON ex.id_examen = n.id_examen
GROUP BY c.id_cours, c.nom_cours
HAVING AVG(n.valeur) < 12;
```

Requête 31 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    c.nom_cours,
    i.date_inscription
FROM Inscription i
JOIN Etudiant e ON i.id_etudiant = e.id_etudiant
JOIN Cours c ON i.id_cours = c.id_cours
WHERE i.annulee = 0
ORDER BY e.nom, i.date_inscription;
```

-- Requête 32 :

```
SELECT DISTINCT
    e.id_etudiant,
    e.nom,
    e.prenom
FROM Etudiant e
JOIN Note n ON e.id_etudiant = n.id_etudiant
JOIN Examen ex ON n.id_examen = ex.id_examen
WHERE ex.nom_examen = 'Examen Informatique 1';
```

-- Requête 33 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    ex.nom_examen,
    n.valeur
FROM Etudiant e
JOIN Note n ON e.id_etudiant = n.id_etudiant
JOIN Examen ex ON n.id_examen = ex.id_examen
JOIN Cours c ON ex.id_cours = c.id_cours
WHERE c.nom_cours = 'Informatique'
ORDER BY e.nom, ex.nom_examen;
```

Requête 34 :

```
SELECT
    ex.id_examen,
    ex.nom_examen,
    ex.date_examen,
    ex.semestre,
    en.nom AS nom_enseignant,
    en.prenom AS prenom_enseignant
FROM Examen ex
JOIN Cours c ON ex.id_cours = c.id_cours
JOIN Enseignant en ON ex.id_enseignant = en.id_enseignant
WHERE c.nom_cours = 'Informatique'
ORDER BY ex.date_examen;
```

-- Requête 35 :

```
SELECT
    c.id_cours,
    c.nom_cours,
    en.id_enseignant,
    en.nom AS nom_enseignant,
    en.prenom AS prenom_enseignant
FROM Cours c
JOIN Enseignant en ON c.id_enseignant = en.id_enseignant
ORDER BY c.nom_cours;
```

-- Requête 36 :

```
SELECT
    en.id_enseignant,
    en.nom,
    en.prenom,
    COUNT(DISTINCT i.id_etudiant) AS nombre_etudiants
FROM Enseignant en
JOIN Cours c ON en.id_enseignant = c.id_enseignant
LEFT JOIN Inscription i ON c.id_cours = i.id_cours AND i.annulee = 0
GROUP BY en.id_enseignant, en.nom, en.prenom
ORDER BY nombre_etudiants DESC;
```

Requête 37 :

```
SELECT
    c.id_cours,
    c.nom_cours
FROM Cours c
LEFT JOIN Inscription i ON c.id_cours = i.id_cours AND i.annulee = 0
WHERE i.id_inscription IS NULL;
```

-- Requête 38 :

```
SELECT
    en.id_enseignant,
    en.nom,
    en.prenom,
    AVG(n.valeur) AS moyenne_notes_etudiants
FROM Enseignant en
JOIN Examen ex ON en.id_enseignant = ex.id_enseignant
JOIN Note n ON ex.id_examen = n.id_examen
GROUP BY en.id_enseignant, en.nom, en.prenom
ORDER BY moyenne_notes_etudiants DESC;
```

-- Requête 39 :

```
SELECT
    e.id_etudiant,
    e.nom,
    e.prenom,
    n.valeur
FROM Etudiant e
JOIN Note n ON e.id_etudiant = n.id_etudiant
JOIN Examen ex ON n.id_examen = ex.id_examen
WHERE ex.nom_examen = 'Examen Économie 1' AND n.valeur < 10;
```

Requête 40 :

```
SELECT
    en.id_enseignant,
    en.nom,
    en.prenom,
    COUNT(c.id_cours) AS nombre_cours
FROM Enseignant en
LEFT JOIN Cours c ON en.id_enseignant = c.id_enseignant
GROUP BY en.id_enseignant, en.nom, en.prenom
ORDER BY nombre_cours DESC;
```

-- Requête 41 :

```
SELECT
    c.id_cours,
    c.nom_cours
FROM Cours c
JOIN Enseignant en ON c.id_enseignant = en.id_enseignant
WHERE en.nom = 'Dupont';
```

Conclusion :

Ce MCD offre une couverture complète des processus universitaires, facilite l'évolution future (ajout de modules, stages, etc.) et pose une base solide pour la génération automatique du schéma relationnel et l'implémentation dans un SGBD.

Bibliographie :

- Utilisé pour la réalisation du MCD :
  https://www.looping-mcd.fr/

- Utilisé pour comprendre les fondamentaux de la construction d'un diagramme ER et ses relations :
  https://www.guru99.com/fr/er-diagram-tutorial-dbms.html

- Employé comme outil principal pour réaliser graphiquement le schéma Entité-Relation:
  https://www.diagrams.net

- Consulté pour approfondir la maîtrise des principes de cardinalité dans un modèle ER:
  https://www.youtube.com/

- Utilisé pour clarifier la représentation et la modélisation des clés étrangères dans le diagramme ER:
  Microsoft Copilot