

## A- Modelisation

### 1- 1. MCD (description texte — entités, attributs essentiels, relations et cardinalités)

#### Entités :

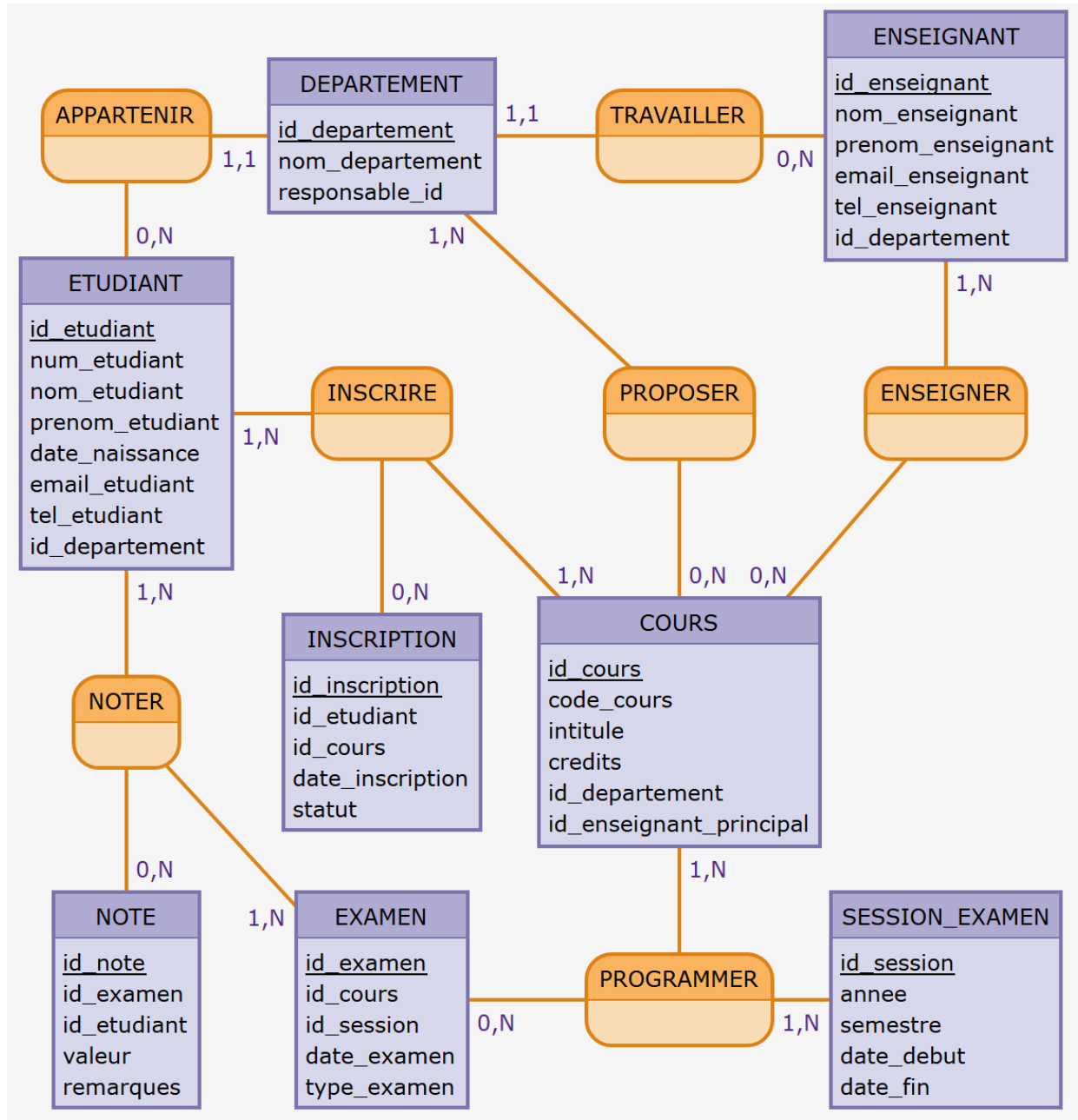
1. ETUDIANT(id\_etudiant, num\_etudiant, nom, prenom, date\_naissance, email, telephone, id\_departement)
2. ENSEIGNANT(id\_enseignant, nom, prenom, email, telephone, id\_departement)
3. DEPARTEMENT(id\_departement, nom\_departement, responsable\_id)
4. COURS(id\_cours, code\_cours, intitule, credits, id\_departement, id\_enseignant\_principal)
5. SESSION\_EXAMEN(id\_session, annee, semestre, date\_debut, date\_fin)
6. INSCRIPTION(id\_inscription, id\_etudiant, id\_cours, date\_inscription, statut)
7. EXAMEN(id\_examen, id\_cours, id\_session, date\_examen, type\_examen)
8. NOTE(id\_note, id\_examen, id\_etudiant, valeur, remarques)

#### Principales relations et cardinalités (résumé) :

- Un **departement** a **plusieurs enseignants** (1,N). Un enseignant travaille dans un departement (N,1).
- Un **departement** propose **plusieurs cours** (1,N). Un cours appartient à un departement (N,1).
- Un **enseignant** peut enseigner **plusieurs cours** (1,N); un cours a au moins un enseignant responsable (N,1).
- Un **etudiant** peut s'inscrire à **plusieurs cours** (1,N) via **INSCRIPTION** ; un cours a **plusieurs etudiants** (N,1).

- Un **cours** peut avoir **plusieurs examens** (contrôle, rattrapage) dans différentes sessions (1,N).
- Un **examen** peut avoir **plusieurs notes** (1,N) — une par étudiant participant. Et une **note** lie un étudiant à un examen.

2 -



Association	Signification	Cardinalités
TRAVAILLER	un enseignant → un département	(N,1)
APPARTENIR	un étudiant → un département	(N,1)
PROPOSER	un département → plusieurs cours	(1,N)
ENSEIGNER	un enseignant → plusieurs cours	(1,N)
INSCRIRE	étudiant ↔ cours via inscription	(1,N) / (1,N)
PROGRAMMER	un cours → examens via sessions	(1,N)
NOTER	une note lie étudiant + examen	(1,N)

3-

La 3NF exige que **toute dépendance fonctionnelle non triviale  $X \rightarrow Y$**  vérifie que :

1. **X est une clé candidate,**  
**OU**
2. **Y est un attribut premier** (c'est-à-dire appartenant à une clé candidate).

**Table ETUDIANT(id\_etudiant, num\_etudiant, nom, prenom, date\_naissance, email, telephone, id\_departement)**

**Dépendance principale :**

**id\_etudiant → (tout le reste)**

Aucune autre DF car :

- Le numéro étudiant n'est pas garanti unique dans la description (même si en pratique oui)
- Les noms/prénoms n'identifient rien d'autre

Pas de dépendance transitive, pas d'attribut dérivé.

**ETUDIANT est en 3NF**

**Table ENSEIGNANT(id\_enseignant, nom, prenom, email, telephone, id\_departement)**

Clé : id\_enseignant

Toutes les DF sont de type :

**id\_enseignant → autres attributs**

Aucune dépendance transitive.

➡ **ENSEIGNANT est en 3NF**

**Table DEPARTEMENT(id\_departement, nom\_departement, responsable\_id)**

Clé : id\_departement

DF :

- id\_departement → nom\_departement
- id\_departement → responsable\_id

responsable\_id n'identifie aucun autre attribut du département → pas de transitivité.

➡ **DEPARTEMENT est en 3NF**

**Table COURS(id\_cours, code\_cours, intitule, credits, id\_departement, id\_enseignant\_principal)**

Clé : id\_cours

DF :

- id\_cours → tout le reste

Potentiel problème :

`id_enseignant_principal` → `nom/prenom/email/telephone`

Mais ces attributs ne sont pas dans la table COURS, donc aucune transitive interne.

→ **COURS est en 3NF**

**Table SESSION\_EXAMEN(id\_session, annee, semestre, date\_debut, date\_fin)**

Clé : **id\_session**

DF :

- `id_session` → (`annee`, `semestre`, `date_debut`, `date_fin`)

Aucune dépendance entre les attributs.

→ **SESSION\_EXAMEN est en 3NF**

**Table INSCRIPTION(id\_etudiant, id\_cours, date\_inscription, statut)**

Clé : (`id_etudiant`, `id_cours`)

DF :

- (`id_etudiant`, `id_cours`) → `date_inscription`
- (`id_etudiant`, `id_cours`) → `statut`

Aucune dépendance :

- partielle,
- transitive,
- redondance sur étudiant ou cours.

→ **INSCRIPTION est en 3NF**

**Table EXAMEN(id\_examen, id\_cours, id\_session, date\_examen, type\_examen)**

Clé : **id\_examen**

DF :

- $\text{id\_examen} \rightarrow \text{autres attributs}$

Aucune transitive interne.

➡ **EXAMEN est en 3NF**

**Table NOTE(id\_note, id\_examen, id\_etudiant, valeur, remarques)**

Si ta clé est **id\_note**, simple :

DF :

- $\text{id\_note} \rightarrow \text{id\_examen, id\_etudiant, valeur, remarques}$

Aucune DF problématique.

Si tu utilisais la clé composite (id\_examen, id\_etudiant), la justification reste OK.

➡ **NOTE est en 3NF**

**Donc modèle relationnel est entièrement en 3NF**

## **Partie B :**

### **SQL - Creation de la base (DDL)**

#### **1. Table DEPARTEMENT**

```
CREATE TABLE DEPARTEMENT (  
  id_departement INT PRIMARY KEY,  
  nom_departement VARCHAR(100) NOT NULL UNIQUE,  
  responsable_id INT  
);
```

#### **2. Table ENSEIGNANT**

```
CREATE TABLE ENSEIGNANT (  
  id_enseignant INT PRIMARY KEY,  
  nom VARCHAR(50) NOT NULL,  
  prenom VARCHAR(50) NOT NULL,  
  email VARCHAR(100) UNIQUE NOT NULL,  
  telephone VARCHAR(20),  
  id_departement INT NOT NULL,
```

```
FOREIGN KEY (id_departement) REFERENCES DEPARTEMENT(id_departement)
);
```

Mise à jour responsable dans DEPARTEMENT (FK après création des deux tables)

```
ALTER TABLE DEPARTEMENT
ADD FOREIGN KEY (responsable_id) REFERENCES ENSEIGNANT(id_enseignant);
```

### **3. Table ETUDIANT**

```
CREATE TABLE ETUDIANT (
  id_etudiant INT PRIMARY KEY,
  num_etudiant VARCHAR(20) NOT NULL UNIQUE,
  nom VARCHAR(50) NOT NULL,
  prenom VARCHAR(50) NOT NULL,
  date_naissance DATE NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  telephone VARCHAR(20),
  id_departement INT NOT NULL,
  FOREIGN KEY (id_departement) REFERENCES DEPARTEMENT(id_departement)
);
```

### **4. Table COURS**

```
CREATE TABLE COURS (
  id_cours INT PRIMARY KEY,
  code_cours VARCHAR(20) NOT NULL UNIQUE,
  intitule VARCHAR(100) NOT NULL,
  credits INT CHECK (credits > 0),
  id_departement INT NOT NULL,
  id_enseignant_principal INT NOT NULL,
  FOREIGN KEY (id_departement) REFERENCES DEPARTEMENT(id_departement),
  FOREIGN KEY (id_enseignant_principal) REFERENCES ENSEIGNANT(id_enseignant)
);
```

### **5. Table SESSION\_EXAMEN**

```
CREATE TABLE SESSION_EXAMEN (
  id_session INT PRIMARY KEY,
  annee INT NOT NULL CHECK (annee >= 2000),
  semestre INT NOT NULL CHECK (semestre IN (1,2)),
  date_debut DATE NOT NULL,
  date_fin DATE NOT NULL
);
```

## 6. Table INSCRIPTION

```
CREATE TABLE INSCRIPTION (  
  id_inscription INT PRIMARY KEY,  
  id_etudiant INT NOT NULL,  
  id_cours INT NOT NULL,  
  date_inscription DATE NOT NULL,  
  statut VARCHAR(20) CHECK (statut IN ('valide','en attente','annule')),  
  FOREIGN KEY (id_etudiant) REFERENCES ETUDIANT(id_etudiant),  
  FOREIGN KEY (id_cours) REFERENCES COURS(id_cours)  
);
```

## 7. Table EXAMEN

```
CREATE TABLE EXAMEN (  
  id_examen INT PRIMARY KEY,  
  id_cours INT NOT NULL,  
  id_session INT NOT NULL,  
  date_examen DATE NOT NULL,  
  type_examen VARCHAR(20) CHECK (type_examen IN ('controle','rattrapage')),  
  FOREIGN KEY (id_cours) REFERENCES COURS(id_cours),  
  FOREIGN KEY (id_session) REFERENCES SESSION_EXAMEN(id_session)  
);
```

## 8. Table NOTE

```
CREATE TABLE NOTE (  
  id_note INT PRIMARY KEY,  
  id_examen INT NOT NULL,  
  id_etudiant INT NOT NULL,  
  valeur DECIMAL(4,2) CHECK (valeur >= 0 AND valeur <= 20),  
  remarques VARCHAR(255),  
  FOREIGN KEY (id_examen) REFERENCES EXAMEN(id_examen),  
  FOREIGN KEY (id_etudiant) REFERENCES ETUDIANT(id_etudiant)  
);
```

Table	Clés primaires	Clés étrangères	UNIQUE	CHECK
DEPARTEMENT	id_departement	responsable_id	nom_departement	responsable_id > 0
ENSEIGNANT	id_enseignant	id_departement	email, telephone	email LIKE '%@%'
ETUDIANT	id_etudiant	id_departement	num_etudiant, email	date_naissance < today
COURS	id_cours	id_departement, id_enseignant_principal	code_cours	credits > 0
SESSION_EXAMEN	id_session	_____	_____	semestre ∈ {1,2}
EXAMEN	id_examen	id_cours, id_session	_____	type_examen contrôlé
INSCRIPTION	id_inscription	id_etudiant, id_cours	(id_etudiant, id_cours)	statut contrôlé
NOTE	id_note	id_examen, id_etudiant	(id_examen, id_etudiant)	$0 \leq \text{valeur} \leq 20$

### 3 - Insérer au moins 5 tuples par table.

#### Table DEPARTEMENT

```
INSERT INTO DEPARTEMENT(id_departement, nom_departement, responsable_id)
VALUES
(1, 'Informatique', NULL),
(2, 'Mathématiques', NULL),
(3, 'Physique', NULL),
(4, 'Chimie', NULL),
(5, 'Électronique', NULL);
```

On met `responsable_id = NULL` pour l'instant, car les enseignants ne sont pas encore insérés.

#### Table ENSEIGNANT

```
INSERT INTO ENSEIGNANT(id_enseignant, nom, prenom, email, telephone, id_departement)
VALUES
(1, 'Durand', 'Marc', 'marc.durand@univ.fr', '0621436578', 1),
(2, 'Leroy', 'Sophie', 'sophie.leroy@univ.fr', '0612680967', 2),
(3, 'Benoît', 'Luc', 'luc.benoit@univ.fr', '0667093467', 1),
(4, 'Martin', 'Alice', 'alice.martin@univ.fr', '0612457853', 3),
(5, 'Loris', 'Hugo', 'hugo.petit@univ.fr', '0616543489', 4);
```

### Mise à jour des responsables de départements

```
UPDATE DEPARTEMENT SET responsable_id = 1 WHERE id_departement = 1;  
UPDATE DEPARTEMENT SET responsable_id = 2 WHERE id_departement = 2;  
UPDATE DEPARTEMENT SET responsable_id = 4 WHERE id_departement = 3;  
UPDATE DEPARTEMENT SET responsable_id = 5 WHERE id_departement = 4;  
UPDATE DEPARTEMENT SET responsable_id = 3 WHERE id_departement = 5;
```

### Table ETUDIANT

```
INSERT INTO ETUDIANT(id_etudiant, num_etudiant, nom, prenom, date_naissance, email,  
telephone, id_departement)  
VALUES  
(1, 'E2024001', 'Kaimoussi', 'Salim', '2004-07-14', 'salim.kaimoussi@etu.fr', '060355484', 1),  
(2, 'E2024002', 'Hassani', 'Nisrine', '2003-11-03', 'nisrine.hassani@etu.fr', '0700000002', 2),  
(3, 'E2024003', 'Diallo', 'Omar', '2001-09-18', 'omar.diallo@etu.fr', '0700000003', 1),  
(4, 'E2024004', 'Nguyen', 'Laura', '2003-01-27', 'laura.nguyen@etu.fr', '0734568784', 3),  
(5, 'E2024005', 'Alaoui', 'yassemine', '2004-05-21', 'yassemine.alaoui@etu.fr', '0700000005', 1);
```

### Table COURS

```
INSERT INTO COURS(id_cours, code_cours, intitule, credits, id_departement,  
id_enseignant_principal)  
VALUES  
(1, 'INF101', 'Algorithmique', 6, 1, 1),  
(2, 'INF102', 'Bases de données', 6, 1, 3),  
(3, 'MAT201', 'Analyse II', 5, 2, 2),  
(4, 'PHY150', 'Mécanique', 4, 3, 4),  
(5, 'CHM110', 'Chimie organique', 5, 4, 5);
```

### Table SESSION\_EXAMEN

```
INSERT INTO SESSION_EXAMEN(id_session, annee, semestre, date_debut, date_fin)  
VALUES  
(1, 2025, 1, '2025-01-10', '2026-01-20'),  
(2, 2025, 2, '2025-06-01', '2026-06-15'),  
(3, 2025, 1, '2025-01-22', '2026-01-29'),  
(4, 2025, 2, '2025-06-20', '2026-06-30'),  
(5, 2025, 1, '2025-02-01', '2026-02-10');
```

### **Table INSCRIPTION**

```
INSERT INTO INSCRIPTION(id_inscription, id_etudiant, id_cours, date_inscription, statut)
VALUES
(1, 1, 1, '2025-01-02', 'valide'),
(2, 1, 2, '2025-01-05', 'valide'),
(3, 2, 3, '2025-01-08', 'valide'),
(4, 3, 1, '2025-01-07', 'valide'),
(5, 4, 4, '2025-01-06', 'valide');
```

### **Table EXAMEN**

```
INSERT INTO EXAMEN(id_examen, id_cours, id_session, date_examen, type_examen)
VALUES
(1, 1, 1, '2025-01-15', 'controle'),
(2, 1, 2, '2026-06-10', 'rattrapage'),
(3, 2, 1, '2025-01-17', 'controle'),
(4, 3, 2, '2026-06-05', 'controle'),
(5, 4, 1, '2025-01-18', 'controle');
```

### **Table NOTE**

```
INSERT INTO note (id_note, id_examen, id_etudiant, valeur, remarques)
VALUES
(1, 1, 1, 14.5, 'Bon travail'),
(2, 2, 2, 12.0, 'Peut mieux faire'),
(3, 3, 3, 16.0, 'Très bien'),
(4, 4, 4, 11.5, 'Correct'),
(5, 5, 5, 17.0, 'Excellent');
```

## **3 - Requetes à realiser :**

### **1. Liste des étudiants (nom, prénom, département)**

```
SELECT e.nom, e.prenom, d.nom_departement
FROM etudiant e
JOIN departement d ON e.id_departement = d.id_departement;
```

### **2 - Cours enseignés par un enseignant donné**

```
SELECT c.code_cours, c.intitule, c.credits
FROM cours c
WHERE c.id_enseignant_principal = ?;
```

### **3. Étudiants inscrits à un cours donné**

```
SELECT et.nom, et.prenom
FROM inscription i
JOIN etudiant et ON i.id_etudiant = et.id_etudiant
WHERE i.id_cours = ?;
```

### **4. Inscriptions d'un étudiant donné**

```
SELECT c.code_cours, c.intitule, i.date_inscription, i.statut
FROM inscription i
JOIN cours c ON i.id_cours = c.id_cours
WHERE i.id_etudiant = ?;
```

### **5. Examens d'un cours (date + type)**

```
SELECT date_examen, type_examen
FROM examen
WHERE id_cours = ?;
```

### **6. Notes obtenues par un étudiant**

```
SELECT c.code_cours, c.intitule, n.valeur, n.remarques
FROM note n
JOIN examen e ON n.id_examen = e.id_examen
JOIN cours c ON e.id_cours = c.id_cours
WHERE n.id_etudiant = ?;
```

### **7. Enseignants d'un département**

```
SELECT nom, prenom, email
FROM enseignant
WHERE id_departement = ?;
```

## 8. Nombre d'étudiants par département

```
SELECT d.nom_departement, COUNT(*) AS nb_etudiants
FROM etudiant e
JOIN departement d ON e.id_departement = d.id_departement
GROUP BY d.nom_departement;
```

## 9. Cours ayant plus de 30 étudiants inscrits

```
SELECT c.id_cours, c.code_cours, c.intitule, COUNT(i.id_etudiant) AS nb_inscriptions
FROM cours c
JOIN inscription i ON c.id_cours = i.id_cours
GROUP BY c.id_cours, c.code_cours, c.intitule
HAVING COUNT(i.id_etudiant) > 30;
```

## 10. Étudiants n'ayant aucune inscription

```
SELECT e.id_etudiant, e.nom, e.prenom
FROM etudiant e
LEFT JOIN inscription i ON e.id_etudiant = i.id_etudiant
WHERE i.id_etudiant IS NULL;
```

## Requêtes avancées

## 11. Moyenne générale d'un étudiant

```
SELECT e.id_etudiant,
       e.nom,
       e.prenom,
       AVG(n.note) AS moyenne_generale
FROM etudiant e
JOIN inscription i ON e.id_etudiant = i.id_etudiant
JOIN examen ex ON i.id_cours = ex.id_cours
JOIN note n ON ex.id_examen = n.id_examen
WHERE e.id_etudiant = 1 -- mettre l'ID de l'étudiant
GROUP BY e.id_etudiant, e.nom, e.prenom;
```

## 12. Moyenne par cours

```
SELECT c.id_cours,  
       c.nom_cours,  
       AVG(n.note) AS moyenne_du_cours  
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;
```

## 13. Classement des étudiants dans un cours (par moyenne décroissante)

```
SELECT e.id_etudiant,  
       e.nom,  
       e.prenom,  
       AVG(n.note) AS moyenne  
FROM etudiant e  
JOIN inscription i ON e.id_etudiant = i.id_etudiant  
JOIN examen ex ON i.id_cours = ex.id_cours  
JOIN note n ON ex.id_examen = n.id_examen  
WHERE i.id_cours = 1 -- mettre l'ID du cours  
GROUP BY e.id_etudiant, e.nom, e.prenom  
ORDER BY moyenne DESC;
```

## 14. Taux de réussite par cours (note $\geq 10$ )

```
SELECT c.id_cours,  
       c.nom_cours,  
       (SUM(CASE WHEN n.note >= 10 THEN 1 ELSE 0 END) * 100.0  
        / COUNT(n.note)) AS taux_reussite  
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;
```

## 15. Top 5 des meilleurs étudiants d'un département

Moyenne générale + tri + limite top 5

```
SELECT e.id_etudiant,  
       e.nom,  
       e.prenom,  
       d.nom_departement,  
       AVG(n.note) AS moyenne_generale  
FROM etudiant e  
JOIN departement d ON e.id_departement = d.id_departement  
JOIN inscription i ON e.id_etudiant = i.id_etudiant  
JOIN examen ex ON i.id_cours = ex.id_cours  
JOIN note n ON ex.id_examen = n.id_examen  
WHERE d.id_departement = 1 -- mettre l'ID du département  
GROUP BY e.id_etudiant, e.nom, e.prenom, d.nom_departement  
ORDER BY moyenne_generale DESC  
LIMIT 5;
```