Liste des Requêtes pour la gestion de la Plateforme Universitaire

1. **Afficher les étudiants qui n'ont pas encore passé d'examens.**

   SELECT e.* FROM etudiant e LEFT JOIN note n ON e.id_etudiant = n.id_etudiant WHERE n.id_note IS NULL;

2. **Calculer le nombre total d'étudiants inscrits à la plateforme.**

   SELECT COUNT(*) AS total_etudiants FROM etudiant;

3. **Trouver les étudiants ayant amélioré leurs notes entre deux semestres consécutifs.**

   SELECT e.id_etudiant, e.nom, e.prenom, AVG(CASE WHEN c.semestre = 'S5' THEN n.note END) AS moyenne_S5, AVG(CASE WHEN c.semestre = 'S6' THEN n.note END) AS moyenne_S6 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 GROUP BY e.id_etudiant, e.nom, e.prenom HAVING moyenne_S6 > moyenne_S5;

4. **Obtenir la moyenne des notes pour chaque cours.**

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

5. **Afficher les détails des enseignants qui encadrent des examens (avec nom de l'examen).**

   SELECT ens.*, ex.nom_examen FROM enseignant ens JOIN examen ex ON ens.id_ens = ex.id_ens;

6. **Afficher le nombre total d'examens organisés pour chaque cours.**

   SELECT c.id_cours, c.nom_cours, COUNT(ex.id_examen) AS nb_examens FROM cours c LEFT JOIN examen ex ON c.id_cours = ex.id_cours GROUP BY c.id_cours, c.nom_cours;

7. **Obtenir la répartition des étudiants par tranche d'âge (moins de 20 ans, 20-30 ans, plus de 30 ans).**

   SELECT CASE WHEN FLOOR(DATEDIFF(CURDATE(), e.date_naissance)/365.25) < 20 THEN 'moins de 20 ans' WHEN FLOOR(DATEDIFF(CURDATE(), e.date_naissance)/365.25) BETWEEN 20 AND 30 THEN '20-30 ans' ELSE 'plus de 30 ans' END AS tranche_age, COUNT(*) AS nombre_etudiants FROM etudiant e GROUP BY tranche_age;

8. **Afficher les étudiants ayant une moyenne générale supérieure à 15/20.**

   SELECT e.id_etudiant, e.nom, e.prenom, AVG(n.note) AS moyenne FROM etudiant e JOIN note n ON e.id_etudiant = n.id_etudiant GROUP BY e.id_etudiant, e.nom, e.prenom HAVING moyenne > 15;

9. **Trouver les enseignants qui ne dispensent aucun cours.**

   SELECT ens.* FROM enseignant ens LEFT JOIN cours c ON ens.id_ens = c.id_ens WHERE c.id_cours IS NULL;

10. **Lister les cours dispensés par un enseignant donné (nom de l'enseignant).**

    SELECT c.* FROM cours c JOIN enseignant e ON c.id_ens = e.id_ens WHERE e.nom = 'Dupont' AND e.prenom = 'Jean';

11. **Obtenir le nombre total d'inscriptions par cours et trier par ordre décroissant.**

    SELECT c.id_cours, c.nom_cours, COUNT(i.id_inscription) AS nb_inscriptions FROM cours c LEFT JOIN inscription i ON c.id_cours = i.id_cours GROUP BY c.id_cours, c.nom_cours ORDER BY nb_inscriptions DESC;

12. Afficher les étudiants qui ont obtenu la meilleure note pour chaque examen.

```
SELECT ex.id_examen, ex.nom_examen, e.nom, e.prenom, n.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.note = ( SELECT MAX(n2.note) FROM note n2 WHERE n2.id_examen = ex.id_examen );
```

13. Lister les inscriptions réalisées après une date donnée.

```
SELECT * FROM inscription WHERE date_inscription > '2025-01-15';
```

14. Obtenir le nombre moyen d'inscriptions par étudiant.

```
SELECT AVG(nb) AS moyenne_inscriptions FROM ( SELECT COUNT(*) AS nb FROM inscription GROUP BY id_etudiant ) t;
```

15. Trouver les inscriptions faites par un étudiant donné (nom et prénom).

```
SELECT i.* FROM inscription i JOIN etudiant e ON i.id_etudiant = e.id_etudiant WHERE e.nom = 'Durand' AND e.prenom = 'Denis';
```

16. Afficher les étudiants inscrits à un cours spécifique (nom du cours donné).

```
SELECT e.* 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 = 'X';
```

17. Lister l'évolution du nombre d'inscriptions par mois pour une année donnée.

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

18. **Obtenir la liste des étudiants avec leur moyenne générale pour chaque cours auquel ils sont inscrits.**

    SELECT e.id_etudiant, e.nom, e.prenom, c.id_cours, c.nom_cours, AVG(n.note) AS moyenne FROM etudiant e JOIN inscription i ON e.id_etudiant = i.id_etudiant JOIN cours c ON i.id_cours = c.id_cours JOIN examen ex ON c.id_cours = ex.id_cours JOIN note n ON ex.id_examen = n.id_examen AND n.id_etudiant = e.id_etudiant GROUP BY e.id_etudiant, c.id_cours;

19. **Afficher les étudiants qui sont inscrits à plus de 3 cours.**

    SELECT e.id_etudiant, e.nom, e.prenom, COUNT(i.id_cours) AS nb_cours FROM etudiant e JOIN inscription i ON e.id_etudiant = i.id_etudiant GROUP BY e.id_etudiant HAVING nb_cours > 3;

20. **Obtenir la liste des enseignants et les cours qu'ils dispensent.**

    SELECT e.id_ens, e.nom, e.prenom, c.id_cours, c.nom_cours FROM enseignant e LEFT JOIN cours c ON e.id_ens = c.id_ens;

21. **Afficher les inscriptions annulées ou supprimées.**

    SELECT * FROM inscription WHERE statut = 'annule';

22. **Trouver l'enseignant qui encadre le plus grand nombre de cours.**

    SELECT e.id_ens, e.nom, e.prenom, COUNT(c.id_cours) AS nb_cours FROM enseignant e JOIN cours c ON e.id_ens = c.id_ens GROUP BY e.id_ens ORDER BY nb_cours DESC LIMIT 1;

23. **Afficher la liste de tous les étudiants avec leur nom, prénom, date de naissance et email.**

    SELECT nom, prenom, date_naissance, email FROM etudiant;

24. **Obtenir la liste des cours et le nombre total d'inscriptions pour chaque cours.**

```
SELECT c.id_cours, c.nom_cours, COUNT(i.id_inscription) AS nb_inscriptions FROM cours c LEFT JOIN inscription i ON c.id_cours = i.id_cours GROUP BY c.id_cours, c.nom_cours;
```

25. **Lister les cours qui ont plus de 50 étudiants inscrits.**

```
SELECT c.id_cours, c.nom_cours, COUNT(DISTINCT i.id_etudiant) AS nb_etudiants FROM cours c JOIN inscription i ON c.id_cours = i.id_cours WHERE i.statut = 'valide' GROUP BY c.id_cours HAVING nb_etudiants > 50;
```

26. **Afficher les cours ayant le taux de réussite le plus élevé.**

```
SELECT c.id_cours, c.nom_cours, COUNT(CASE WHEN n.note >= 10 THEN 1 END) AS nb_reussites, COUNT(n.note) AS nb_total, COUNT(CASE WHEN n.note >= 10 THEN 1 END) / COUNT(n.note) * 100 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 HAVING nb_total > 0 ORDER BY taux_reussite DESC;
```

27. **Afficher le nombre d'inscriptions annulées par mois.**

```
SELECT MONTH(date_inscription) AS mois, COUNT(*) AS nb_annulees FROM inscription WHERE statut = 'annule' GROUP BY mois;
```

28. **Trouver les étudiants qui ont été inscrits à tous les cours offerts par un enseignant donné.**

```
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 JOIN enseignant prof ON c.id_ens = prof.id_ens WHERE prof.nom = 'Dupont' AND prof.prenom = 'Jean' GROUP BY e.id_etudiant HAVING COUNT(DISTINCT c.id_cours) = ( SELECT COUNT(*) FROM cours WHERE id_ens = prof.id_ens );
```

29. **Lister les étudiants, leurs cours, et leurs enseignants pour chaque inscription.**

```
SELECT i.id_inscription, e.nom AS etudiant_nom, e.prenom AS
etudiant_prenom, c.nom_cours, ens.nom AS enseignant_nom, ens.prenom AS
enseignant_prenom 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 ens ON
c.id_ens = ens.id_ens;
```

30. **Obtenir la liste des cours où la moyenne des notes est inférieure à 12/20.**

```
SELECT c.id_cours, c.nom_cours, AVG(n.note) AS moyenne 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 HAVING moyenne < 12;
```

31. **Afficher la date d'inscription de chaque étudiant pour chaque cours.**

```
SELECT e.nom, e.prenom, c.nom_cours, i.date_inscription, i.statut FROM
inscription i JOIN etudiant e ON i.id_etudiant = e.id_etudiant JOIN cours c ON
i.id_cours = c.id_cours;
```

32. **Trouver les étudiants qui ont passé un examen spécifique.**

```
SELECT DISTINCT e.* 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 = 'X';
```

33. **Afficher les notes obtenues par les étudiants pour un cours donné.**

```
SELECT e.nom, e.prenom, ex.nom_examen, n.note FROM note n JOIN etudiant
e ON n.id_etudiant = e.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 =
'Programmation Java';
```

34. **Lister tous les examens programmés pour un cours donné.**

```
SELECT ex.* FROM examen ex JOIN cours c ON ex.id_cours = c.id_cours WHERE
c.nom_cours = 'X';
```

35. **Obtenir la liste de tous les cours disponibles et leur enseignant responsable.**

```
SELECT c.nom_cours , ens.nom, ens.prenom FROM cours c JOIN enseignant ens ON c.id_ens = ens.id_ens;
```

36. **Lister les enseignants et le nombre d'étudiants inscrits dans leurs cours.**

```
SELECT e.id_ens, e.nom, e.prenom, COUNT(DISTINCT i.id_etudiant) AS nb_etudiants FROM enseignant e LEFT JOIN cours c ON e.id_ens = c.id_ens LEFT JOIN inscription i ON c.id_cours = i.id_cours GROUP BY e.id_ens;
```

37. **Afficher les cours qui n'ont pas encore d'inscriptions.**

```
SELECT c.* FROM cours c LEFT JOIN inscription i ON c.id_cours = i.id_cours WHERE i.id_inscription IS NULL;
```

38. **Afficher les statistiques de performance des étudiants par enseignant (moyenne des notes par enseignant).**

```
SELECT e.id_ens, e.nom, e.prenom, AVG(n.note) AS moyenne FROM enseignant e JOIN examen ex ON e.id_ens = ex.id_ens JOIN note n ON ex.id_examen = n.id_examen GROUP BY e.id_ens;
```

39. **Trouver les étudiants ayant échoué (note inférieure à 10/20) à un examen donné.**

```
SELECT e.nom, e.prenom, n.note FROM note n JOIN etudiant e ON n.id_etudiant = e.id_etudiant JOIN examen ex ON n.id_examen = ex.id_examen WHERE ex.nom_examen = 'Examen 1 - Programmation Java' AND n.note < 10;
```

40. **Afficher le nombre total de cours dispensés par chaque enseignant.**

```
SELECT e.id_ens, e.nom, e.prenom, COUNT(c.id_cours) AS nb_cours FROM enseignant e LEFT JOIN cours c ON e.id_ens = c.id_ens GROUP BY e.id_ens;
```

41. Afficher les cours dispensés par un enseignant donné.

```
SELECT c.id_cours, c.nom_cours, e.nom, e.prenom FROM cours c JOIN
enseignant e ON c.id_ens = e.id_ens WHERE e.nom = 'Dupont' AND e.prenom =
'Jean';
```