TD : COMMANDES SQL

1 Lister les noms des guildes du jeu.
```
SELECT nom FROM guildes ORDER BY nom;
```

2 Lister les joueurs (prénom, nom) triés par niveau décroissant.
```
SELECT prenom, nom FROM joueurs ORDER BY niveau DESC;
```

3 Lister les types de PNJ disponibles (sans doublons).
```
SELECT DISTINCT pnj.type FROM pnj
```

4 Lister les PNJ de type « marchand » dont le niveau est > 20.
```
SELECT pnj.nom FROM pnj WHERE pnj.niveau>20
```

5 Afficher la durée moyenne (en minutes) des participations réussies par quête.
```
SELECT AVG(duree_effective_min) FROM participations WHERE succes='1'
```

6 Afficher le titre des quêtes et le nom de leur zone.
```
SELECT q.titre, z.nom AS zone FROM quetes AS q
JOIN zones AS z ON q.zone_id=z.zone_id
```

7 Trouver les quêtes dont le niveau requis est entre 8 et 15 (inclus).
```
SELECT quetes.titre FROM quetes
WHERE quetes.niveau_requis BETWEEN 8 AND 15
```

8 Compter le nombre de joueurs par guilde.
```
SELECT g.nom, COUNT(*) AS nombre_de_joueurs
FROM guildes AS g JOIN joueurs AS j ON j.guilde_id = g.guilde_id
GROUP BY g.guilde_id
```

9 Donner le titre des 2 quêtes les plus tentées (par nombre de participations), avec leur total.
```
SELECT q.titre AS titre, COUNT(*) AS nombre
FROM quetes AS q JOIN participations AS p ON q.quete_id=p.quete_id
GROUP BY q.quete_id
ORDER BY COUNT(*) DESC LIMIT 2
```

10 Donner le titre des quêtes qui ont été tentées plus de 1 fois avec leur nombre de tentatives.
```
SELECT q.titre AS titre, COUNT(*) AS nombre
FROM quetes AS q JOIN participations AS p ON q.quete_id=p.quete_id
GROUP BY q.quete_id HAVING COUNT(*) > 1
```

11 Pour chaque zone, afficher leur nom, le niveau_min et nombre de quêtes disponibles.
```
SELECT z.nom, z.niveau_min, COUNT(*) AS nbr_quetes
FROM zones AS z JOIN quetes AS q ON z.zone_id=q.zone_id
GROUP BY q.zone_id
```

12 Lister les id des participations entre deux dates (ex. date de début du 2025-09-01 au 2025-09-30, avec BETWEEN).

```
SELECT participation_id FROM participations
WHERE DATE(date_debut) BETWEEN '2025-09-01' AND '2025-09-30'
ORDER BY date_debut;
```

13 Afficher les id des 2 dernières participations.

```
SELECT participation_id
FROM participations
ORDER BY date_debut DESC LIMIT 2;
```

14 Afficher le titre des quêtes avec leur durée effective moyenne.

```
SELECT q.titre, AVG(p.duree_effective_min) AS duree_moyenne
FROM quetes AS q JOIN participations AS p ON q.quete_id = p.quete_id
GROUP BY q.quete_id
```

15 Afficher le titre de la quête avec la durée effective moyenne maximale (globalement).

```
SELECT q.titre, AVG(p.duree_effective_min) AS duree_moyenne
FROM quetes AS q JOIN participations AS p ON q.quete_id = p.quete_id
GROUP BY q.quete_id ORDER BY duree_moyenne DESC LIMIT 1
```

16 Lister les (prenom, nom) des joueurs dont le niveau est inférieur au niveau_min de la zone « Désert Rouge ».

```
SELECT j.prenom, j.nom, j.niveau FROM joueurs AS j
WHERE niveau < (SELECT niveau_min FROM zones WHERE nom="Désert Rouge")
```

17 Lister le titre des quêtes dont la durée estimée minimale dépasse la durée effective moyenne observée.

```
SELECT q.titre, q.duree_estimee_min, AVG(p.duree_effective_min) AS
duree_moy_obs
FROM quetes q
JOIN participations p ON p.quete_id = q.quete_id
GROUP BY q.quete_id
HAVING q.duree_estimee_min > AVG(p.duree_effective_min)
```

18 Lister les couples [joueur(prenom, nom), titre de la quête, nombre de tentatives] pour lesquels toutes les participations ont été des échecs (HAVING).

```
SELECT j.prenom, j.nom, q.titre, COUNT(*) AS tentatives
FROM participations p
JOIN joueurs j ON j.joueur_id = p.joueur_id
JOIN quetes q ON q.quete_id = p.quete_id
GROUP BY j.joueur_id
HAVING SUM(p.succes = 1) = 0
```

19 Pour chaque nom de guilde, afficher le niveau moyen des joueurs et ne garder que celles > 25.

```
SELECT g.nom, AVG(j.niveau) AS niveau_moyen
FROM guildes AS g JOIN joueurs AS j ON g.guilde_id=j.guilde_id
GROUP BY g.nom HAVING niveau_moyen > 25
```

20 Lister le nom des guildes au niveau moyen des joueurs > moyenne globale des joueurs.
```
SELECT g.nom, AVG(j.niveau)
FROM guildes AS g JOIN joueurs AS j ON j.guilde_id = g.guilde_id
GROUP BY g.guilde_id
HAVING AVG(j.niveau)>(SELECT AVG(joueurs.niveau) FROM joueurs)
```

21 Lister les zones dont le niveau requis moyen des quêtes est supérieur à la moyenne globale des niveaux requis.
```
SELECT z.nom AS zone, AVG(q.niveau_requis) AS niv_requis_moy
FROM zones z JOIN quetes q ON q.zone_id = z.zone_id
GROUP BY z.zone_id
HAVING AVG(q.niveau_requis) >(SELECT AVG(niveau_requis) FROM quetes);
```

22 Lister les quêtes situées dans des zones de niveau minimal est ≥ 10 en utilisant l'opérateur IN.
```
SELECT q.titre
FROM quetes q
WHERE q.zone_id IN (SELECT zone_id FROM zones WHERE niveau_min >= 10);
```

23 Lister les PNJ dont le type n'est « marchand » ni « forgeron » en utilisant l'opérateur NOT IN.
```
SELECT nom
FROM pnj
WHERE type NOT IN ('marchand','forgeron');
```

24 Lister les joueurs dont la guilde n'est la guilde d'aucun joueur en utilisant l'opérateur NOT IN et une sous-requête dans le NOT IN. Que remarquez-vous ?
```
SELECT j.prenom, j.nom
FROM joueurs j
WHERE j.guilde_id NOT IN (SELECT guilde_id FROM joueurs);
```

25 Modifier votre requête en utilisant NOT EXISTS pour répondre à la question précédente.
```
SELECT j.prenom, j.nom
FROM joueurs j WHERE NOT EXISTS(SELECT 1 FROM guildes AS g WHERE
g.guilde_id=j.guilde_id)
```

26 Lister les paires des id des joueurs qui appartiennent à une même guilde en utilisant une auto-jointure.
```
SELECT j1.joueur_id AS id_joueur_A, j2.joueur_id AS id_joueur_B
FROM joueurs AS j1
JOIN joueurs AS j2 ON j1.guilde_id = j2.guilde_id
AND j1.joueur_id < j2.joueur_id
```

27 Même question mais en affichant le nom de la guilde à laquelle appartiennent les paires de joueurs.
```
SELECT j1.joueur_id AS id_joueur_A, j2.joueur_id AS id_joueur_B, g.nom
FROM joueurs AS j1
JOIN joueurs AS j2 ON j1.guilde_id = j2.guilde_id
JOIN guildes AS g ON j1.guilde_id = g.guilde_id
AND j1.joueur_id > j2.joueur_id
```