```
EXPLAIN ANALYZE
SELECT *
FROM etudiants
inner join logs_connexions on id_etu = id_etu_fk
inner join cours on id_cours = id_cours_fk
WHERE matiere = 'Deep Learning 1';

EXPLAIN ANALYZE
SELECT *
FROM etudiants
inner join logs_connexions on id_etu = id_etu_fk
inner join cours on id_cours = id_cours_fk
WHERE id_cours = 43;
```

Ici on fait la même requête ce qui change c'est que le where tape sur une colonne non indexé (en 1er) et indexé (en 2e). La 2e requete est deux fois plus rapide, 0.5s contre 1.3s.

```
EXPLAIN ANALYZE
SELECT c.matiere, COUNT(i.id_etu_fk) AS nb_inscrits
FROM cours c
LEFT JOIN inscriptions i ON i.id_etu_fk IN (
    SELECT id_etu FROM etudiants
)
GROUP BY c.matiere
ORDER BY nb_inscrits desc;

EXPLAIN ANALYZE
SELECT c.matiere, COUNT(id_etu) AS nb_inscrits
FROM cours c
LEFT JOIN logs_connexions lc ON lc.id_cours_fk = c.id_cours
LEFT JOIN etudiants e ON e.id_etu = lc.id_etu_fk
GROUP BY c.matiere
ORDER BY nb_inscrits desc;
```

Ici on a 2 requêtes complexes, la 2e ne fait pas appel à une sous-requête ce qui a amélioré de beaucoup la requête. 15s contre 890s.

```
EXPLAIN analyze
select *
from inscriptions
where date_insc = '2025-08-26';

Index Scan using inscriptions_pkey on inscriptions  (cost=0.43..8.45 rows=1 width=13)
(actual time=0.120..0.120 rows=0 loops=1)
  Index Cond: (date_insc = '2025-08-26 00:00:00'::timestamp without time zone)
Planning Time: 0.107 ms
Execution Time: 0.152 ms
```

On voit ici qu'il utilise l'index scan d'où son faible temps d'exécution

```
EXPLAIN ANALYZE
SELECT *
FROM etudiants
WHERE nom = 'Martin';
```

```
Seq Scan on etudiants  (cost=0.00..3654.00 rows=992 width=18) (actual
time=21.729..21.730 rows=0 loops=1)
  Filter: ((nom)::text = 'Martin'::text)
  Rows Removed by Filter: 200000
Planning Time: 0.070 ms
Execution Time: 21.758 ms
```

Ici on voit qu’il utilise un seq scan ce qui met du temps