**SQL Assignment**

**Task Title:** SQL Query Practice – dvdrental Database  
**Assignee:** Zoya Hussain  
**Priority:** Medium  
**Due Date:** (Friday, 23rd May)

#### **Objective**

Build fluency in SQL by writing, verifying, and documenting queries that progress from basics (SELECT … WHERE) to advanced techniques (window functions, CTEs, indexing) on the **dvdrental** sample database.

#### **Setup Instructions**

1. **Clone the starter repo** provided in Slack (sql-practice-<your-name>).
2. Install **PostgreSQL 15+** (or run docker-compose up inside the repo).
3. Restore the dump:

  
psql -U postgres -f dvdrental.dump

1. Verify with \dt that 15 tables are present.
2. Create a clean branch named feature/sql-practice.

#### **Work Instructions**

| **Tier** | **File to edit** | **Instruction** | **Tests** |
| --- | --- | --- | --- |
| **T0 – Warm-up** | challenges/t0\_select.sql | Complete three SELECT queries; run pytest -k t0 | Must pass |
| **T1 – Filters** | challenges/t1\_filters.sql | Add WHERE, BETWEEN, LIKE queries | pytest -k t1 |
| **T2 – Aggregation** | challenges/t2\_agg.sql | Write COUNT/SUM/AVG with GROUP BY | pytest -k t2 |
| **T3 – Joins** | challenges/t3\_joins.sql | Write INNER & LEFT joins to meet specs | pytest -k t3 |
| **T4 – CTE/Subquery** | challenges/t4\_cte.sql | Use WITH queries to answer prompts | pytest -k t4 |
| **T5 – Window Functions** | challenges/t5\_window.sql | Implement ROW\_NUMBER, LAG, running totals | pytest -k t5 |
| **T6 – Set Ops & Pivot** | challenges/t6\_set.sql | UNION/INTERSECT, pivot ratings with CASE | pytest -k t6 |
| **T7 – Performance** | challenges/t7\_perf.sql | Identify slow query, add index, paste EXPLAIN ANALYZEbefore & after | pytest -k t7 |

**General rules**

* Use **meaningful aliases**; avoid SELECT \*.
* Prepend each query with a single-line comment explaining intent.
* Keep each query self-contained—no temp tables.
* Commit after each tier with message feat(sql): complete T<n>.

#### **Deliverables**

1. **Completed SQL files** in challenges/ folder.
2. **tests/ all green** in local run *and* GitHub Actions pipeline.
3. **docs/performance-notes.md** showing runtime improvement ≥ 10× for Tier 7, with both EXPLAIN ANALYZE plans pasted.
4. Short **demo (≤ 5 min)** during team call:
   * One tricky query walkthrough
   * Key learning & difficulties.

#### **Acceptance Criteria**

1. All 30 + pytest cases pass in CI; badge shows green.
2. A new teammate can follow README.md and run tests in < 30 min.
3. Performance note clearly states *why* the chosen index helps.
4. Git history shows logical, incremental commits (≥ 8).
5. Reviewer signs off that SQL style is clear and commented.

#### **Reference Cheatsheet**

* \dt – list tables · \d+ <table> – describe table
* EXPLAIN (ANALYZE, BUFFERS) <query>; – detailed plan
* Window syntax: SUM(amount) OVER (PARTITION BY store\_id ORDER BY payment\_date)
* Pivot example:

  
SELECT

SUM(CASE WHEN rating='G' THEN 1 END) AS g\_count,

SUM(CASE WHEN rating='PG' THEN 1 END) AS pg\_count,

SUM(CASE WHEN rating='PG-13' THEN 1 END) AS pg13\_count

FROM film;

