**1. Implement Optimistic Concurrency Control for Tournament Updates**

📌 **Problem:** Two admins attempt to change **the start date** for the same tournament at the same time.

✅ **Task:**

* Add a **version column** to Tournaments.
* Implement **version-based optimistic concurrency control**
* Ensure that only one change is successful when two concurrent admins try to update.

Løsning

**Fil: db\_queries.py**

**2. Implement Pessimistic Concurrency Control for Match Updates**

**📌 Problem: Two admins attempt to update the same match result at the same time. Ensure only one update happens at a time.**

**✅ Task:**

* **Implement pessimistic locking using SELECT ... FOR UPDATE.**
* **Ensure only one admin can update match results at a time.**

**Løsning fil:**

**PessimisticConcurrency.py**

**locked\_No\_update.py**

**3. Handle Transactions for Tournament Registrations**

**📌 Problem: Ensure atomicity when registering a player in a tournament. If any part of the transaction fails, rollback all changes.**

**✅ Task:**

* **If registration is successful, insert a record into Tournament\_Registrations and update player ranking.**
* **If the tournament is full, rollback the transaction.**

**Løsning fil:**

**Transactions\_Tournament\_Registrations.py**

**4.  Implement a Stored Procedure for Safe Ranking Updates**

**📌 Problem: A player’s ranking should increase after winning a match. Ensure concurrent updates do not cause inconsistencies.**

**✅ Task:**

* **Create a stored procedure that updates player ranking.**
* **Use pessimistic locking to prevent simultaneous updates.**

**5. Implement Concurrency Control of Your Own Choice for Tournament Registration**

📌 **Problem:** Two players attempt to register for the same tournament at the same time. If the **max\_players** limit is reached, one should be rejected.

**✅ Task:**

* Ensure that only one registration is successful when two concurrent users try to register.

**Løsning fil:**

Concurrency\_Tournament\_Registration.py