Project Documentation: Gaming Tournament Database

By Kasey Lightheart

1.Purpose of the Database

This gaming tournament management database is designed to store and manage information related to gaming tournaments, users, teams, matches, and results. It enables easy and efficient tracking of schedules for teams, and prize distributions while also maintaining historical records of matches and player participation.

2.Database ERD Schema

A screenshot of a computer

Description automatically generated

3.Sample Questions and SQL Query.

**Good Question:**

“*Which teams placed in the top 3 of any tournament?”*

**Generated SQL Query:**

SELECT t.tournament\_name, tm.team\_id, tm.placement

FROM TournamentResults tm

JOIN Tournament t ON tm.tournament\_id = t.tournament\_id

WHERE tm.placement <= 3

ORDER BY t.tournament\_name, tm.placement;

**Results:**

('Battle Royale Championship', 1, 1)

('Fantasy Quest Finals', 2, 1)

**Question it got tripped up on:**

"*Which users have sent messages but never participated in a tournament?"*

**Generated SQL Query:**

SELECT u.user\_id, u.username

FROM User u

LEFT JOIN Message m ON u.user\_id = m.user\_id

LEFT JOIN TournamentResults tr ON u.user\_id = tr.team\_id

WHERE m.user\_id IS NOT NULL AND tr.team\_id IS NULL;

**Results:**

(3, 'NoobMaster')

**Response:**

Basically, the joins GPT tried to make here were incorrect, because based on the populated data from these tournaments, no one who sent a message didn’t participate in a tournament. The proper SQL should have joined User user\_id with TeamMembership user\_id and then TeamMembership team\_id with TournamentResults team\_id where tournament\_id IS NULL like so:

SELECT DISTINCT u.user\_id, u.username

FROM User u

JOIN Message m ON u.user\_id = m.user\_id

LEFT JOIN TeamMembership tm ON u.user\_id = tm.user\_id

LEFT JOIN TournamentResults tr ON tm.team\_id = tr.team\_id

WHERE tr.tournament\_id IS NULL;

4. **Prompting Strategies Used**

**Zero-Shot Prompting**

Zero-shot prompting involves providing GPT with only the database schema and the user's natural language query, without any examples of how similar queries should be translated into SQL. This approach was straightforward and required minimal setup. GPT was generally able to generate correct SQL queries for simple and well-structured questions, such as:

*"Which teams placed in the top 3 of any tournament?"*

However, **zero-shot** prompting struggled with more complex queries involving multiple relationships across tables. For example:

*"Which users have sent messages but never participated in a tournament?"*

GPT incorrectly attempted to join the User table directly with TournamentResults, using the condition JOIN TournamentResults tr ON u.user\_id = tr.team\_id, which does not correctly link users to their team participation. Instead, it should have used TeamMembership to relate users to teams before checking TournamentResults.