CPSC 304 Project Cover Page

Milestone #: 4

Date: 2024-11-29

Group Number: 16

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Nazia Edroos	20010476	d3e6q	edroos.nazia@gmail.com
Prajna Nayak	78725462	j8s1q	prajnapn36@gmail.com
Rachel Wang	71451769	i6a7p	rachelwang0432@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your email address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Repository Link:
https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_d3e6q_i6a7p_j8s1q.git

SQL SCRIPT

- Can be found in file InitDatabaseSchema.sql, InitDatabaseData.sql

A short description of the final project, and what it accomplished.

Populated our own database and implemented SQL queries in the backend for users to be able to find information in our Mario Kart Competition Database. Users can track the races they have participated in with the respective character, kart and power ups used. For each race that a user participated in, they can also view their placement, which competition it was part of, what tracks they raced on and what hazards were on these tracks. We hope that this database can help players, professional or casual, understand their playing style and improve their skills.

A description of how your final schema differed from the schema you turned in. If the final schema differed, explain why.

- Hazard and Track Schema (TrackContainsHazard → TrackHazard), zone attribute; we felt that the frequency of the hazard on a track was a more valuable descriptor for users of the database compared to what zones the hazards appear in
- Track Schema, trackShortcuts attribute; we changed the data type from VARCHAR to INT as we felt that the number of shortcuts on a track was a better piece of information

A list of all SQL queries used to satisfy the rubric items and where each query can be found in the code (file name and line number(s)).

• For SQL queries 2.1.7 through 2.1.10 inclusive, include a copy of your SQL query

and a maximum of 1-2 sentences describing what that query does. You can embed

this in your above list of queries. You don’t need to include the output of the query.

2.1.1 INSERT: appService.js line 161

2.1.2 UPDATE : appService.js lines 175-181

2.1.3 DELETE: appService.js line 202

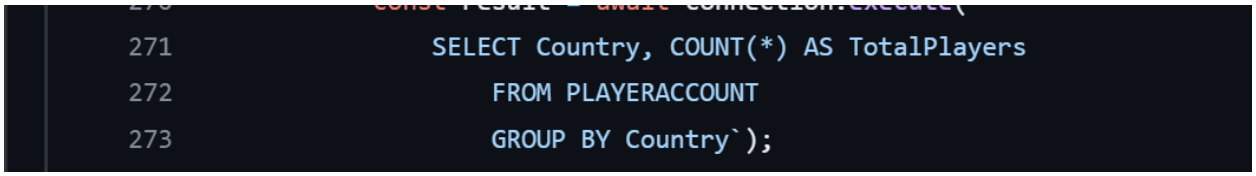
2.1.4 SELECTION: appService.js line 379

2.1.5 PROJECTION: appService.js line 233

2.1.6 JOIN: appService.js lines 250-254

2.1.7 AGGREGATION with GROUP BY: appService.js

- groupByAllCountriesTable() - lines 271-273
    - Uses COUNT aggregation to count the number of players from each country. Groups players by country and returns the count of players in each country.



- groupByUserInputCountriesTable(countrycode) - lines 285-288
    - Uses COUNT aggregation to find the number of players from a user-specified country. Filters players by the provided country code and returns the count of players from that country.

2.1.8 AGGREGATION with HAVING: appService.js lines 447-448

- groupByHaving(minParticipants) - lines 447 - 458:
    - Uses COUNT aggregation to count the number of players in each competition. Gets minimum participants input from the user and returns all competition Ids with number of players equal to or greater than the input.

```
446             const query = `
447                 SELECT competitionID, COUNT(playerID) AS ParticipantCount
448                 FROM PlayerRace
449                 GROUP BY competitionID
450                 HAVING COUNT(playerID) >= :minParticipants
```

2.1.9 NESTED AGGREGATION with GROUP BY:

- appService.js
  - MIN: Lines 360-362
  - MAX: lines 363-366
- Based on user input, either:
  - Group by Hazard; finds the highest/lowest value from the average/count of the number of tracks the hazard appears on
  - Group by Track; finds the highest/lowest value from the average/count of the number of hazards that appears on each track

```
358             let statement;
359             if (query == "min") {
360                 statement = `SELECT *
361                             FROM temp
362                             WHERE "Number" = (SELECT MIN("Number") FROM temp)`;
363             } else if (query == "max") {
364                 statement = `SELECT *
365                             FROM temp
366                             WHERE "Number" = (SELECT MAX("Number") FROM temp)`;
367             }
368
```

2.1.10 DIVISION: appService.js lines 418-429

```
417             const query = `
418                SELECT p.*
419                 FROM PLAYERACCOUNT p
420                 WHERE NOT EXISTS (
421                     -- Check if there's any competition that the player has not competed in
422                     SELECT 1
423                     FROM CompetitionInformation c
424                     WHERE NOT EXISTS (
425                         -- Check if the player participated in the competition
426                         SELECT 1
427                         FROM PlayerRace pr
428                         WHERE pr.playerID = p.playerID
429                         AND pr.competitionID = c.competitionID
```

- Find all players who have participated in every Mario Kart Competition.