University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ____4____

Date: __Nov. 28th, 2024___

Group Number: ____123___

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Sara Zhang	60959509	o9j2b	zhangxiyu100@gmail.com
Koda Tootoosis	47941331	r8j6r	kljtootoosis@gmail.com
Yufei Ren	36267672	b4b3b	xixiryf@126.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 e-mail 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

Summary of the project:

The database is to model a voting system of a competition show, involving singers and bands. It has performances that are matches, each match is formed by two groups that contain bands and singers as the performers. The groups then perform a song from the songs in the database. Judges vote on which groups they think are the best ones, and the audience votes on individual singers that they like.

Differences in the schema:

Changed the table name Group to Performer_Group because “group” is a reserved keyword in SQL.

Changed for better clarity

- Changed match_date to mdate to ensure non-duplication with naming
- Changed num_votes to number_votes for SINGER_R1
- Changed performerID to winnerID for Match_Winner

Changed due to conflicts with sql commands

- Changed name to audience_name for Audience_Info
- Changed name to judge_name for Judge
- Changed date to mdate for Match_Addr
- Changed location to match_location for Match_Cap

Changed because of needing UNIQUE for a tuple that is being referenced by a Foreign Key

- Added UNIQUE to mdate for Match_Date
- Added UNIQUE to birth_date for Singer_R1
- Added UNIQUE to ticket_type for Audience_Info
- Added UNIQUE to match_location for Match_Addr

List of SQL queries:

2.1

SQL query:

```
INSERT INTO DEMOTABLE (id, name) VALUES (:id, :name)
```

File name and line number(s):
appService.js, Line: 262

2.2

SQL query:

```
`UPDATE Performer SET performer_name = :performer_name, debut_year = :debut_year, num_fans = :num_fans, groupID = :groupID WHERE performerID = :performerID`
```

File name and line number(s):
appService.js Line 291-314

2.3
SQL query:
`DELETE FROM Performer WHERE ${condition}`

File name and line number(s):
appService.js 327

2.4
SQL query:
SELECT * FROM Performer WHERE ${condition}

File name and line number(s):
appService.js Line342

2.5
SQL query:
`SELECT ${columns} FROM Performer`

File name and line number(s):
appService.js 357

2.6
SQL query:
```
`SELECT DISTINCT g.song_name
 FROM Performer p, Performer_Group g
 WHERE performerID = :performerID and p.groupID = g.groupID`
```

File name and line number(s):
appService.js Line 372-375

2.7
SQL query:
SELECT groupID, MIN(num_fans) AS min_fan FROM Performer GROUP BY groupID

Description of what this query does(1-2 sentences):
In every group, let's find out the min number of fans of performers in that group.

File name and line number(s):
appService.js Line 372:388

2.8

SQL query:
`SELECT groupID, SUM(num_fans) AS fans
FROM Performer
GROUP BY groupID
HAVING sum(num_fans) >= :minFans

Description of what this query does(1-2 sentences):
“Find the summed number of fans per group which are above or equal to a user inputted minimum amount of fans”

File name and line number(s):
appService.js Line 403-407

2.9
SQL query:
subQuery = `SELECT ${generalSign}(num_fans) FROM Performer`;
`SELECT P.${group_by}, ${select} FROM Performer P GROUP BY P.${group_by} HAVING ${havingSign}(P.num_fans) ${havingConstraint} (${subQuery})`;

Description of what this query does(1-2 sentences):
“ for each debut year, find the max/avg/min of the number of fans of the group of performers who are larger/equal/smaller than the max/avg/min of all the performers.”

File name and line number(s): appService.js, line 420-421

2.10
SQL query:
```
SELECT p.debut_year
FROM Performer p
WHERE NOT EXISTS (
   SELECT pg.groupID
   FROM Performer_Group pg
   MINUS
   SELECT p2.groupID
   FROM Performer p2
   WHERE p2.debut_year = p.debut_year
)
GROUP BY p.debut_year;
```

Description of what this query does(1-2 sentences):
Let's find out the debut years when every group has at least one performer who debuted in that year.

File name and line number(s):

appService.js, Line 436:437