CPSC 304 Project Cover Page

Milestone #: 4
Date: November 29, 2024
Group Number: 93

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Sabrina Lou	17539495	t1e8c	sabrinajwlou@gmail.com
Rohan Shukla	80120785	c7a8v	shuklarohan@live.com
Freddi Li	51907897	f5q5r	winnifredsli38@gmail.com

Project Description
Our project is a public transportation system manager, used within the transportation management sector to manage their own system. It focuses on the operations, logistics, and staff/client management within a public transit system, based on the public transport available in the Metro Vancouver area. This includes managing routes, vehicles, stations and stops, drivers, riders, and fare payment.

On the routes page, users can view and filter routes, stops on a particular bus route, and update the scheduled time or route position of a particular bus stop on a route.
On the client page, users can add, remove, and view clients, view fares, and obtain insights about clients' use of scanners and fares.
On the employee/vehicle page, users can view employees and assigned vehicles and find drivers on selected routes.

Final Schema Changes
One schema change was adding a new UNIQUE attribute to BusRouteStopsAt(stop_id, rid, route_pos), so it became:

BusRouteStopsAt(stop_id, rid, sched_time, route_pos)
PK: (stop_id, rid)
FKs: stop_id, rid
UNIQUE: sched_time

- Sched_time represents the time that the route is supposed to reach the stop, which can be unique to avoid buses arriving at a stop at the same time.
- I needed to make this change to align with the UPDATE query requirement for updating a UNIQUE attribute.

SQL Queries
List all SQL queries with file name and line number(s) it can be found on. For marked queries (hard coded ones), include 1–2 sentences describing what the query does.

<u>INSERT</u>
appService.js lines(203-215)

<u>UPDATE</u>
appService.js line (372 - 382) and (384-394)

<u>DELETE</u>
appService.js lines(217-228)

<u>Selection</u>
appService.js lines(291-322)

<u>Projection</u>
appService.js lines(115-124)

<u>Join</u>
appService.js line (271-280)

<u>Aggregation with GROUP BY (**explain**)</u>
appService.js (347 - 357)

Aggregation with HAVING (**explain**)
appService.js (106-113)
Query:

```
SELECT compass_id, SUM(price)
FROM PaidFares1 p1, PaidFares2 p2
WHERE p1.fare_type = p2.fare_type
GROUP BY compass_id
HAVING count(*) > 1
```

Explanation: Find the total amount of money spent for all clients who have paid for more than one fare.

Nested aggregation with GROUP BY (**explain**)
appService.js line 284
Query:

```
WITH ScannerClientCounts AS (
        SELECT scan_id,
                COUNT(DISTINCT compass_id)
                AS client_count
        FROM ValidateFare
        GROUP BY scan_id)
SELECT scan_id, client_count
FROM ScannerClientCounts
WHERE client_count = (
                SELECT MAX(client_count)
                FROM ScannerClientCounts)
```

Explanation: Finds the scanner with the maximum number of distinct clients who have scanned their fare.

Division (**explain**)

appService.js(97-104)
Query:

```
SELECT r.compass_id
FROM RIDER1 r
WHERE NOT EXISTS (SELECT s.scan_id
                  FROM ScannerHas s
                  WHERE NOT EXISTS (SELECT v.compass_id
                                    FROM ValidateFare v
                                    WHERE v.compass_id = r.compass_id AND
                                v.scan_id = s.scan_id))
```

Explanation: Find all clients who have used every scanner

Note on SQL Script tables:

- can't model participation constraints using Null restrictions, did not implement assertions