# CPSC 304

# Milestone #4

# Date: November 25th, 2022

# Group Number: 72

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Tom Henley | 40054793 | u6i2b | tomhenleyyy@gmail.com |
| Aviva Mei | 74065350 | j7x9t | aviivameii@gmail.com |
| Emily Chu | 26625426 | s3y2b | im.mlechu@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 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**

**REPOSITORY URL:**
https://github.students.cs.ubc.ca/CPSC304-2022W-T1/project_j7x9t_s3y2b_u6i2b

**Deliverables**
Commit the deliverables below to the CPSC 304 provided repository and submit a link to your repository on Canvas.

1. Your completed cover page, as usual.
2. ~~A single SQL script that can be used to create all the tables and data in the database. If you are using multiple scripts while developing, ensure you concatenate them and hand in only a SINGLE SQL script. [done]~~
    a. ~~Non-trivial size: make sure that your queries have some non-trivial answers (e.g., division for only 2 products is trivial) and the same idea with aggregation (Group By), in general, because you need to have a reasonable number of groups), and you need to have some groups that have more than one row.~~
    b. ~~b. This SQL script should be runnable as is (i.e., if we transferred this file to the undergrad servers, we should be able to run it as is without further tweaking).~~
3. A PDF file (this document) containing:
    a. A short description of the final project, and what it accomplished.
    b. A description of how your final schema differed from the schema you turned in.
        i. If the final schema differed, explain why. Note that turning in a final schema that's different from what you planned is fine, we just want to know what changed and why.
    c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run.
    d. A list of all SQL queries used. For SQL query requirements, check the rubric listed on Canvas for Milestone 4.
    e. Screenshots of the sample output of the queries using the GUI (for example, you can show what data is in your table before you run the query, and then show another screenshot after running the query, from some kind of GUI input like a button).
        i. You need only to include screenshots for the required queries – if you implemented more than what was required, screenshots are not needed for those extra queries.
4. Lastly, include a README.txt file if there's anything you want to add that's not included in your PDF file.

# Single SQL Script:

Found in project_j7x9t_s3y2b_u6i2b/304-app/sql/**reset-autogen.sql**

# Description of final project:

Our project allows users to view different information about a transit system, including:

- The bus model information for a specific bus given the bus ID

- The max capacity of bus models grouped by their fuel type
- The max capacity of bus models grouped by their fuel type where there is more than one of that bus model
- The max capacity of bus models grouped by their fuel type where the max capacity is larger than the average capacity of all bus models

Our project also allows users to add, update, and delete compass taps

## Description of final schema differences:

We only made minor changes that simply resulted from typing when creating the tables:

- We changed the order of the attributes in each schema
- We changed the naming convention for the attributes from capitalizing the first letter every word with no space to only lowercase with underscores between words

## Schema:

- Bus(**route_number: CHAR(3)**, **route_name: CHAR(50),** id CHAR(10), **model: CHAR(20),** license_plate: CHAR(6))
- Driver(id: CHAR(8), first_name: CHAR(30), last_name: CHAR(30))
- Skytrain(**route_number**: **CHAR(3), route_name**: **CHAR(50),** id: CHAR(6), **model**: **CHAR(25)**)
- SkytrainStation(**stopid**: **CHAR(10),** name: CHAR(50), platforms: INT)
- BusStop(**stopid: CHAR(10)**, name: CHAR(50))
- Stop(id: CHAR(10), postcode: CHAR(6), **city: CHAR(20)**)
- Zone(zone_number: INT, city: CHAR(20))
- CompassTap(card_id: CHAR(20), time: INT, **stop: CHAR(50)**)
- DriverAssignment(**driver_id: CHAR(8)**, **bus_id: CHAR(10)**)
- AvailableStop( **route_number: CHAR(3), route_name: CHAR(50)**, **stop: CHAR(20)**)
- BusModel(Name: CHAR(20), Cost: INT, Capacity: INT, FuelType: CHAR(6), Cost: INT, PurchaseCost: INT, OperatingCost: INT)
- BusModel1(name: CHAR(20), capacity: INT, fuel_type: CHAR(6), **purchase_cost: INT, operating_cost: INT**)
- BusModel2(purchase_cost: INT, operating_cost: INT, cost: INT)
- SkytrainModel1(name: CHAR(20), capacity: INT, cars: INT, **purchase_cost: INT**, **operating_cost: INT**)

- SkytrainModel2(purchase_cost: INT, operating_cost: INT, cost)
- Route1(**route_number: CHAR(3),** route_name: CHAR(50), distance: REAL, **Origin: CHAR(10)**, **Destination: CHAR(10)**)
- Route2(route_number:CHAR(3), bus_route_type: CHAR(20), rail_type: CHAR(20))

## Data After Insert:

## SQL queries used:

### INSERT Operation

### DELETE Operation

### UPDATE Operation

### Selection

### Projection

### Join

```
SELECT b.id, b.model, b.license_plate, bm.capacity, bm.fuel_type
FROM Bus b, BusModel1 bm
WHERE b.model = bm.name and b.id = (user input)
```

Found in bus-models.php

### Division

### Aggregation with Group By

```
SELECT Max(capacity), fuel_type
FROM BusModels1
GROUP BY fuel_type
```

Found in groupBy.php

### Aggregation with Having

```
SELECT Max(capacity), fuel_type
FROM BusModels1
GROUP BY fuel_type
HAVING Count(*) > 1
```

Found in having.php

### Nested Aggregation with Group By

```
SELECT Max(capacity), fuel_type
FROM BusModels1
GROUP BY fuel_type
HAVING Max(capacity) > (SELECT Avg(capacity) FROM BusModel1)
```

Found in nestedAggregation.php

## Sample Output:

### INSERT Operation

### DELETE Operation

### UPDATE Operation

### Selection

### Projection

## Join

< Go home

### Bus model finder

Existing bus IDs: 18022, 9409, 19027, 9660

Bus ID: 18022 select

SELECT b.id, b.model, b.license_plate, bm.capacity, bm.fuel_type FROM Bus b, BusModel1 bm WHERE b.model = bm.name and b.id = 18022

**Join Table**

| id | license plate | model | model capacity | model fuel type |
|---|---|---|---|---|
| 18022 | NFI XDE60 | NG5745 | 120 | D-E |

## Division

## Aggregation with Group By

< Go home

# Find the max capacity of bus models by fuel type

SELECT Max(capacity), fuel_type
FROM BusModels1
GROUP BY fuel_type

Submit

**Group By Table**

| Fuel Type | Max Capactiy |
|---|---|
| Diesel | 70 |
| D-E | 120 |
| N Gas | 70 |
| B-E | 70 |

## Aggregation with Having

< Go home

### Find the max capacity of bus models by fuel type that have more than 1

```
SELECT Max(capacity), fuel_type
FROM BusModels1
GROUP BY fuel_type
HAVING Count(*) > 1
```

Submit

**Group By Table**

| Fuel Type | Max Capactiy |
|---|---|
| D-E | 120 |

## Nested Aggregation with Group By

< Go home

### Find the max capacity of bus models by fuel type that are larger than the average capacity of all bus models

```
SELECT Max(capacity), fuel_type
FROM BusModels1
GROUP BY fuel_type
HAVING Max(capacity) > (SELECT Avg(capacity) FROM BusModel1)
```

Submit

**Group By Table**

| Fuel Type | Max Capactiy |
|---|---|
| D-E | 120 |