This project was actually a survey by [**Obukofe Uririoghene,**  he reached out to me on twitter and sent me the link to a survey he was conducting using SQL.](https://twitter.com/obukofejoe)

The first task was to create a normalized schema of 11 tables and import the dataset. For this, he provided the queries to run, along with the instructions to follow, which can be found [here](https://obukofe.notion.site/obukofe/SQL-Survey-501f61a843ab4a60879e60601eedeaac).

After data validation, which is step 4 of the project, I started with the main business tasks.

This project involves a managing director of the Lagos Metropolitan BRT service, “Jane”, who is having a bit of trouble managing her database. And also needs help answering the following questions with data from her company’s database.

The questions were as follows;

1. \*\*Arrival Location Rankings\*\*. Jane noticed that some arrival locations stand out more than the others and she intends to find out where her passengers head to the most. She needs a table of the top 5 arrival locations sorted in descending order. Can you help her with that?

2. \*\*Top Performing Drivers\*\*. Jane is keen on employee retention and she’s planning on rewarding her top drivers. She needs to know the top 10 performing drivers based on the number of trips they’ve handled. For this, it would be best if we had a table containing their full names, their NIN’s and the number of trips they’ve handled. Let’s help Jane make that work!

3. \*\*Frequent Passengers\*\*. Jane is a wonderful boss, and she loves her customers. She intends to reward her top 10 customers by offering 50% discounts on their next trips. The problem however is that she doesn’t exactly know which passengers are her most frequent. She needs a table containing the passenger’s full name, their email address and the number of trips they’ve taken sorted in descending order.

4. \*\*Driver - Vehicle Pairings\*\*. Jane lost track of her drivers and their paired vehicles. She needs your help! She wants to know the current driver - vehicle pairings. To help her, let’s create a table containing the full names of the driver and the buses they are paired to. She’d also like to see some details of the vehicle such as the plate number and the model.

5. \*\*Highest Issue Dates\*\*. Jane has been impressed by your work so far but she needs one more thing. She wants to know the issue dates where her customers had the highest number of card issues for trips payment. She wants the list sorted by top 5 issue dates in descending order and also the number of cards issued. Let’s help her with that.

I was also required to provide “open ended queries as you explore more on the database. The queries can be any query of your choice. Think of it as extra recommendations to Jane as she intends to find the answers to more questions in the future regarding her company.”

I started with the original business questions.

**--1. top 5 arrival locations**

SELECT arrival\_location, COUNT(\*)

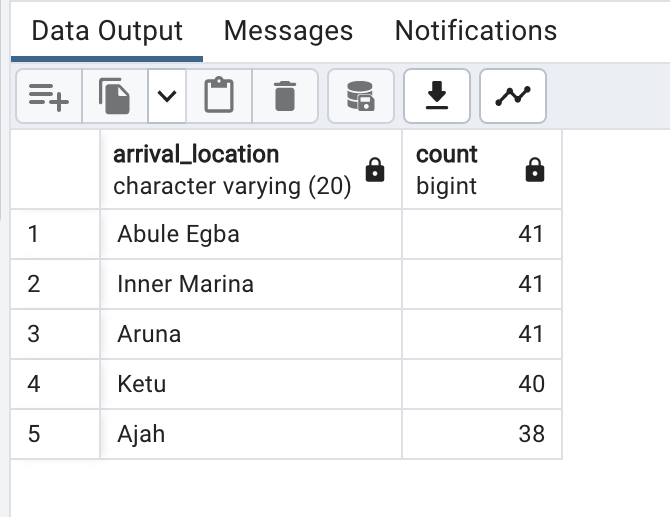
FROM brt.trips

GROUP BY arrival\_location

ORDER BY COUNT(\*) DESC

LIMIT 5;

\*\*For this, it was pretty straight forward, the data was in the trips table, so I used the COUNT function to gather the number of times each location appeared in the arrival\_location column, and ordered it in descending order of the counts, then limited it by 5, to only show the 5 highest values.



**--2. top performing drivers**

ALTER TABLE brt.drivers

ADD COLUMN full\_name text;

UPDATE brt.drivers

SET full\_name = concat(first\_name, ' ', last\_name);

SELECT drivers.full\_name,

drivers.national\_identity\_number AS NIN,

COUNT(trips.driver\_id) AS total\_trips

FROM brt.drivers AS drivers

INNER JOIN brt.trips AS trips

ON drivers.id = trips.driver\_id

GROUP BY drivers.full\_name, drivers.national\_identity\_number

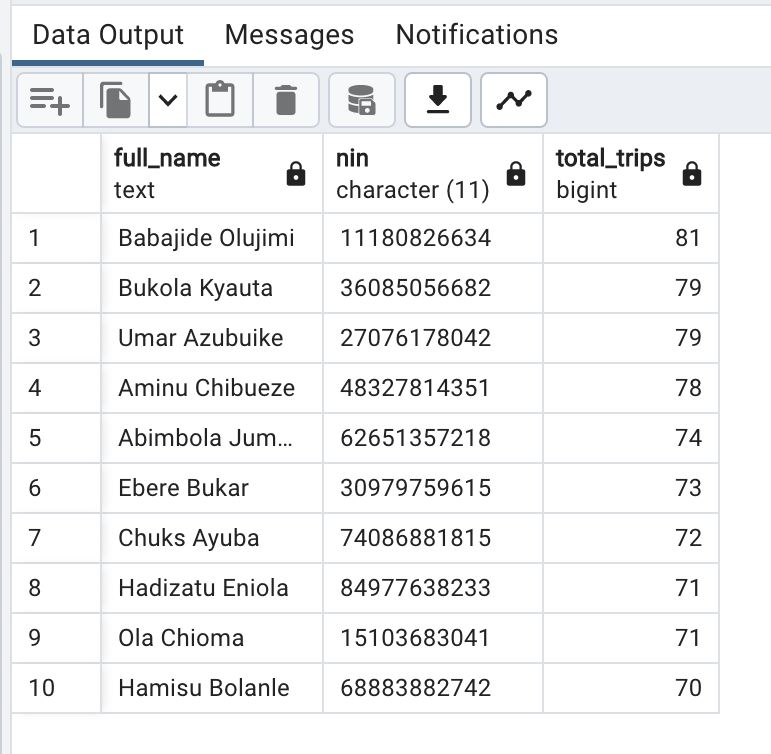
ORDER BY total\_trips DESC

LIMIT 10;

\*\*Here, the requirement was to select the top performing drivers’ full names, NIN, and show the number of trips they have handled. We were asked to provide the drivers’ full names, so I created a new column for their full names, and use the concat() function on the first and last name columns to update their full names.

The required data were in two different tables, so I used INNER JOIN to join the “trips” and “drivers” tables on the condition that the drivers’ id and trips’ driver\_id were the same.

I then selected the required columns, counting the “driver\_id” column for the trips as “total\_trips” to see how many times a driver went on a trip, then I ordered them by total trips in descending order and limited them by 10, to see only the top 10 performing drivers.



**--3. frequent passengers**

ALTER TABLE brt.passengers

ADD COLUMN full\_name text;

UPDATE brt.passengers

SET full\_name = concat(first\_name, ' ', last\_name);

SELECT passengers.full\_name, passengers.email\_address, COUNT(passenger\_trips.passenger\_id) AS total\_trips

FROM brt.passengers AS passengers

INNER JOIN brt.passenger\_trips AS passenger\_trips

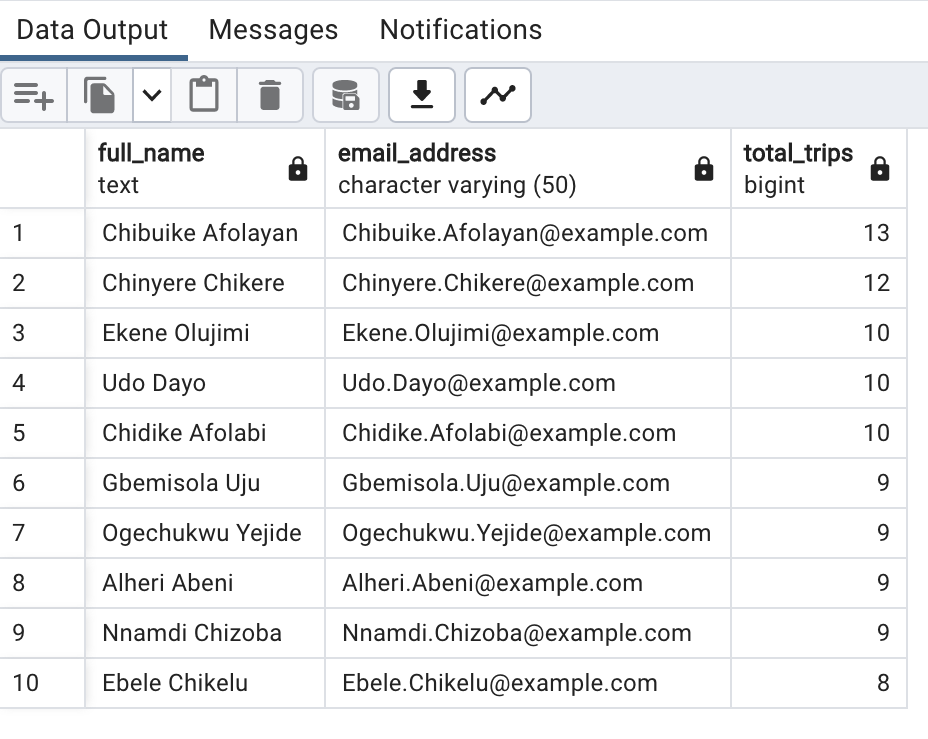
ON passengers.id = passenger\_trips.passenger\_id

GROUP BY passengers.full\_name, passengers.email\_address

ORDER BY total\_trips DESC

LIMIT 10;

\*\*In this question, the task was to select the full names and emails of the top 10 passengers, as in the previous question, I created a full name column for the passengers and updated it, then selected the information needed from both tables, using the COUNT() function again to select the top 10 appearing passenger IDs as total\_trips and ordered them by the top 10 in descending order.



--4.driver - vehicle pairings

SELECT drivers.full\_name as driver\_name,

logs.driver\_id,

vehicles.id as vehicle\_id,

vehicles.model, vehicles.plate\_number

FROM brt.drivers as drivers

JOIN brt.driver\_vehicle\_logs AS logs

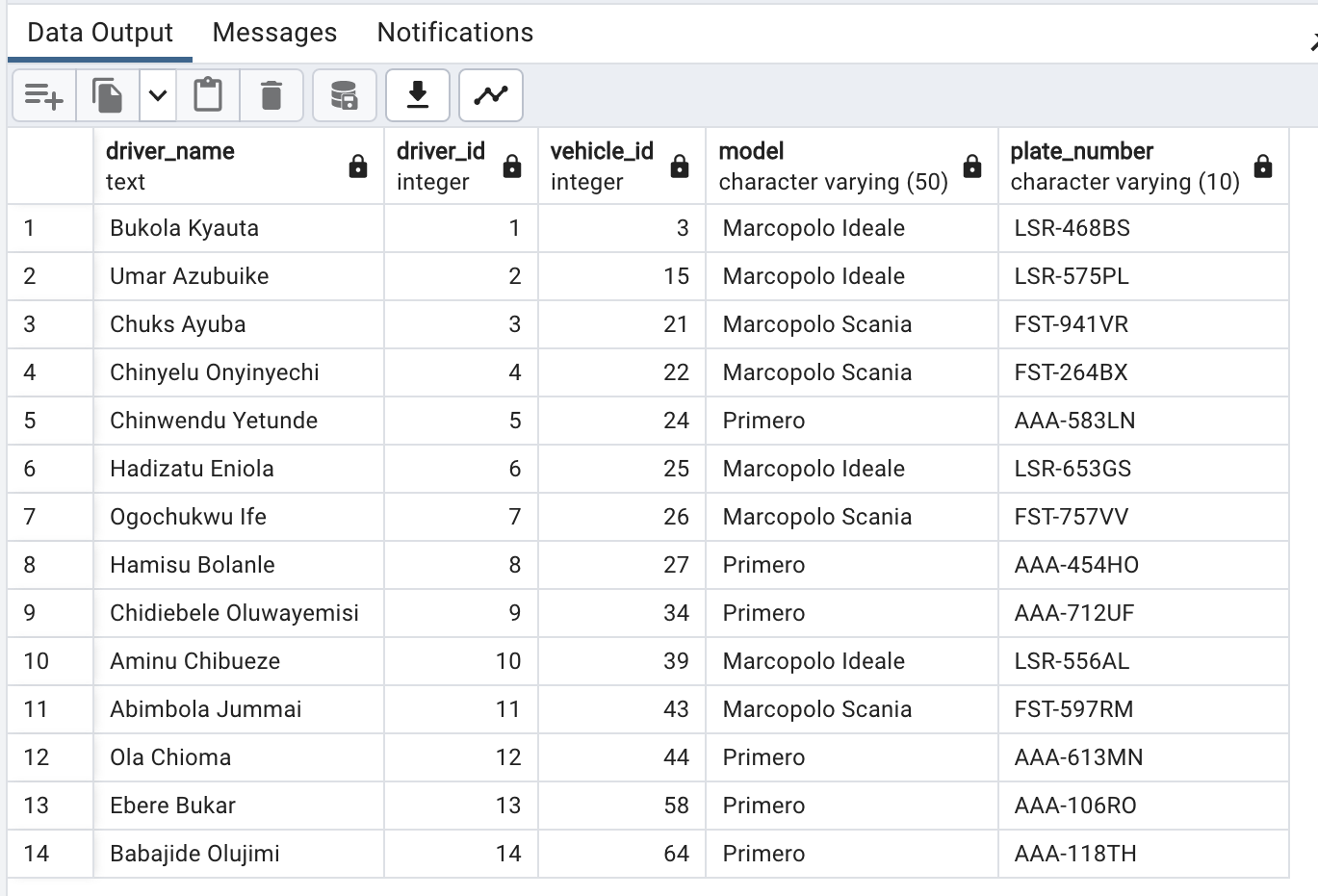
ON drivers.id = logs.driver\_id

JOIN brt.vehicles as vehicles

ON logs.vehicle\_id = vehicles.id;

\*\*Here, the task was to find what drivers were paired to each vehicle, providing their full names, driver id, their vehicle id, the vehicle model, and its plate number.

The information I needed for this was in 3 different tables, so I used two JOIN statements to draw information from the “drivers”, “driver\_vehicle\_logs”, and the “vehicles” tables.



--5.highest issue dates

SELECT DISTINCT (issue\_date), COUNT (\*) as cards\_issued

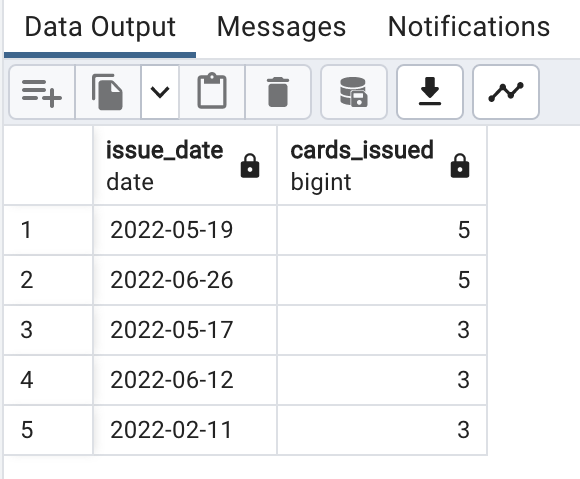
FROM brt.passenger\_cards

GROUP BY issue\_date

ORDER BY COUNT (\*) DESC

LIMIT 5

\*\*The task here was to select the dates where the most cards were issued, as well as the amount of cards issued on that date. The query here was similar to that of question 1; select issue\_date column and count the times each date appeared, order it by the count in descending order and limit it to the top 5 dates.



That was it for the original business questions, next, I had to come up with further questions that could be useful for Jane to have further business insights for her company. After scanning the schema and the table contents, I came up with the following questions;

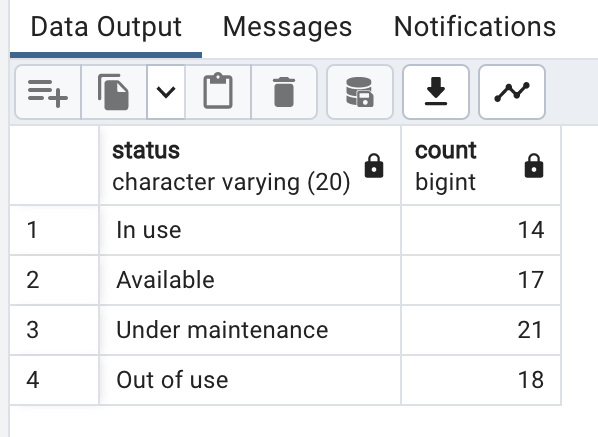
-**-1. no of vehicles in use, available, under maintenance and out of use**

SELECT status, COUNT (\*)

FROM brt.vehicles

GROUP BY status

\*\*This query selected the “status” column from the “vehicles” table, and counted how many times each status appeared.



**--2. top 5 departure locations**

SELECT departure\_location, COUNT(\*)

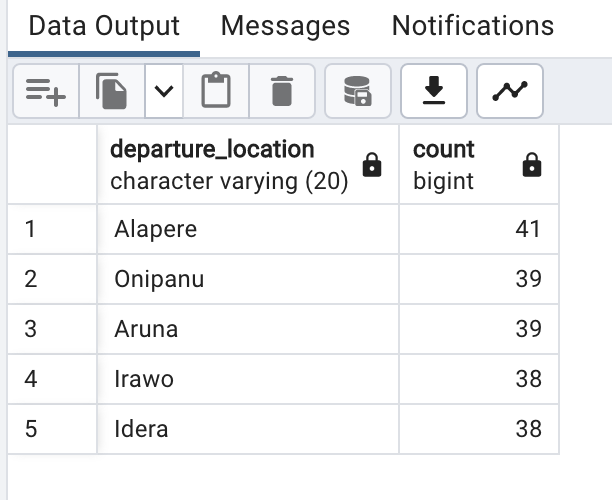
FROM brt.trips

GROUP BY departure\_location

ORDER BY COUNT(\*) DESC

LIMIT 5;

\*\*I ran a query similar to the arrival location one in the original business question, these are the results



**--3. drivers with licenses to be renewed within one year (current date is feb 14, 2023)**

SELECT license.expiry\_date, license.driver\_id, drivers.full\_name

FROM brt.license as license

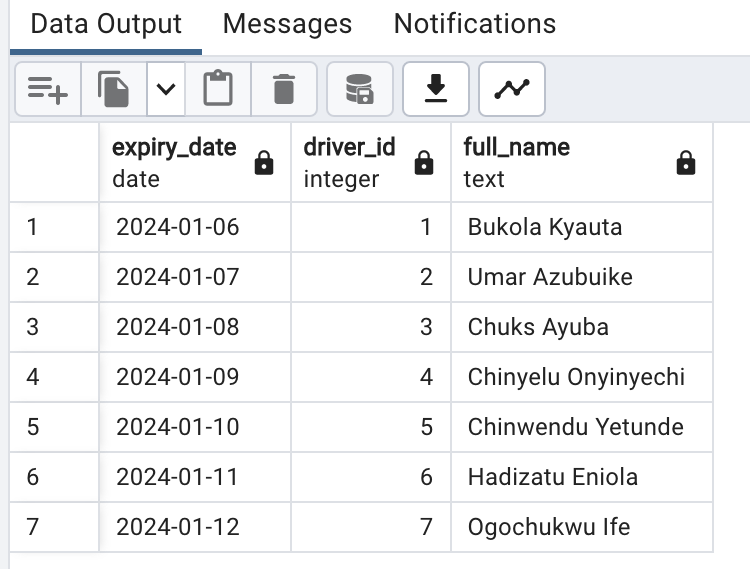
JOIN brt.drivers as drivers

ON license.driver\_id = drivers.id

WHERE license.expiry\_date BETWEEN CURRENT\_DATE AND CURRENT\_DATE + INTERVAL '1 year';

\*\*For this, I used a JOIN statement to draw data from the “license” and “drivers” tables, to show drivers’ full names, IDs, and the expiry date.

I also used a WHERE statement to filter the expiry\_date column to select the dates between the current date(February 14th, 2023) and one year from that date, using the “+ INTERVAL ‘1 year’ clause’.



**--4. drivers with the biggest buses (trip\_capacity)**

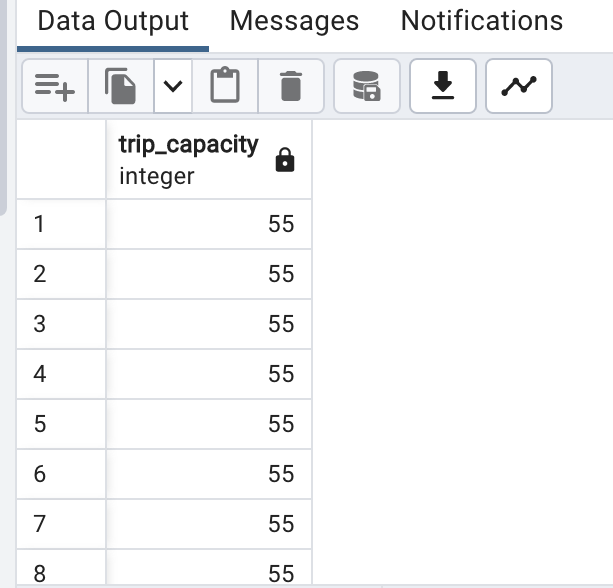
**--viewing highest trip capacity**

SELECT trip\_capacity

FROM brt.trips

ORDER BY trip\_capacity DESC

\*\*I first ran this query to determine what the highest capacities were for the buses



**--viewing drivers details**

SELECT drivers.full\_name as driver\_name,

drivers.id,

trips.trip\_capacity

FROM brt.drivers as drivers

JOIN brt.trips as trips

ON drivers.id = trips.driver\_id

WHERE trip\_capacity = 55

\*\*Then I used this JOIN query to select the buses with the highest capacity, linking them to their drivers information.



**--5. 3 lowest performing drivers**

SELECT drivers.full\_name, drivers.email\_address, COUNT(trips.driver\_id) AS total\_trips

FROM brt.drivers AS drivers

INNER JOIN brt.trips AS trips

ON drivers.id = trips.driver\_id

GROUP BY drivers.full\_name, drivers.email\_address

ORDER BY total\_trips

LIMIT 3;

\*\*This query was simply changing the order from descending to ascending for the query I ran in the original business question number 2, but I decided to limit it to the lowest 3 performing drivers instead, changing it to LIMIT 10. I also decided that their email addresses would be more useful information, to contact them and discuss ways to move forward with regards to their performances.

