Project: Car Rental Database

Your company, Apasaja Pte Ltd, has been commissioned to design and implement a database application to record information about car rental operation for the PinjaMobil Pte Ltd. The application stores and manages historical information about the customers and cars.

The application stores and manages information about customers. A customer is identified by their NRIC number. Some customer may have driving license. The application also stores a name of the customer, the date of birth, as well as car preference. The car preference will be based on a brand and model of the car as stored in the database.

The application also stores and manages information about the cars. A specific car is identified by their license plate with the color of the car recorded. The current scheme for license plate starts with letter S followed by two letters, four digits, and the final checksum letter (see: Wikipedia). The brand and the model of each car are also recorded, which we call as the make of the car.

To accommodate future purchases, the application should allow recording of the make of the car even if we have currently no car with the given brand and model. This is also important as some customer may have a preference for a car make that the company does not have yet. There are also additional information about the capacity of the car that depends on the make of the car. As there is no modification of the car, all cars with the same make will have the same capacity.

The application tracks which customers rented which cars. The record of the start date and end date are recorded for each rental. We will use the notation [start, end] to indicate the interval from start to end *inclusive* of both dates. Given two interval [s1, e1] and [s2, e2], we will still treat the following two cases as overlapping interval: (i) e1 = s2 and (ii) e2 = s1. This is because the interval we are using is *inclusive* of both dates. Obviously, a car already rented in the interval [s1, e1] cannot be rented again in the interval [s2, e2] if the interval overlaps.

The application also tracks all passenger information of a car that are being rented. If a car is not currently being rented, there should be no passenger. For simplicity, the passenger will ride for the entire duration of the rented car. So if a car is being rented in the interval [s, e], then any passenger of this car will ride in the car for the entire interval [s, e].

Note that the customer renting need not be one of the passengers. In other words, a customer may initiate a rent for another passenger. The number of passenger must be smaller than or equal to the capacity of the car depending on their make. Furthermore, one *or more* of the passenger must have a driving license. Similar to how a car cannot be rented on overlapping dates, a customer cannot be a passenger on two different cars being rented with overlapping dates.

Finally, the application stores and manages only historical records and does not manage future events. For our purpose, given the new year of 2025, we will only test on dates of up to 2024. You do not have to make an explicit check for the date.

References

[1] Iso 8601 date and time format. https://www.iso.org/iso-8601-date-and-time-format.html. Visited on 31 January 2024.

[2] Iso week date. https://en.wikipedia.org/wiki/ISO_week_date. Visited on 31 January 2024.

[3] Mockaroo - random data generator and api mocking tool. www.mockaroo.com. Visited on 31 January 2024.

[4] National registration identity card. https://en.wikipedia.org/wiki/National_Registration_Identity_Card. Visited on 31 January 2024.

[5] Vehicle registration plates of singapore. https://en.wikipedia.org/wiki/Vehicle_registration_plates_of_Singapore. Visited on 31 January 2024.

1. (3 points) Creating and Populating Tables with Constraints

   (a) (3 points) DDL

   Write the SQL Data Definition Language (DDL) code, `CREATE TABLE` with integrity constraints, to create the necessary table(s) for storing the data required by the application. Ensure all necessary data can be accommodated, and avoid catering for unnecessary, inconsistent, or unnecessarily redundant data. Propagate updates to maintain referential integrity. Only propagate deletions where it make sense.

   You are advised to first list down all the constraints as a list. Make sure that the constraint is as clear as possible. You may want to split any complex constraints into multiple simpler constraints. While you may not be able to fully enforce complex constraints, you may still be able enforce the simpler sub-constraints.

   (b) (0 points) DML

   You may find more information about the structure of Singapore NRIC from Wikipedia. A similar information about the checksum for vehicle registration plate (*i.e., license plate*) can also be found on Wikipedia. You may use them to generate *realistic data*.

   Other random data can be found from Mockaroo. There is a car make in Mockaroo but it is closer to brand in our terminology. Although technically the model year matters for capacity, we will simplify the problem by ignoring this information.

   Use this data to write SQL Data Manipulation Language (DML) code to populate the database.

   You are recommended to use ISO 8601 format for date as also recommended by PostgreSQL. You may find additional information regarding dates operation on PostgreSQL documentation page. It has many built-in support for operation on dates.

Submission

- Canvas Submission: "`Assignments > Questions 1`" (*one file per project group*)
- **Files**
  - SQL DDL: `schema.sql`

    Ensure that `schema.sql` can be executed on a fresh database.
  - SQL DML: `data.sql`

    Note: Submit two separate files. Ensure the name is correct before submission (i.e., in your computer). Canvas may perform renaming, that is normal and we will accept Canvas renaming scheme.