**select** uid, *COUNT*(books.tripid) **from** books **join** trips t **on** books.tripid = t.tripid **and** t.starttime **BETWEEN** '2018-01-03' **AND** '2019-01-03' **group by** uid **ORDER BY** *COUNT*(books.tripid);

This query selects passenger rank and score from time period Jan 3 2018 to Jan 3 2019. Score was determined by count of his trip in a time period. This query first select trip from trips table between Jan 3 2018 and Jan 3 2019 and inner join with passenger table. Group by operations groups joined entries by passenger id then do a count of trips for each driver id group to compute score.

**select** uid, *COUNT*(leads.tripid) **from** leads **join** trips **on** leads.tripid = trips.tripid **and** trips.starttime **between** '2018-01-03' **and** '2019-01-03' **group by** uid **order by** *count*(leads.tripid);

This query selects driver rank and score from time period Jan 3 2018 to Jan 3 2019. Score was determined by count of his trip in a time period. This query first select trip from trips table between Jan 3 2018 and Jan 3 2019 and inner join with leads table on trip id. Group by operations groups joined entries by driver id then do a count of trips for each driver id group to compute score.

**select** \* **from** trips **join** hasstops **h on** trips.tripid = **h**.tripid **and h**.stopname = 'montreal' **and** starttime **between** '2019-02-14' **and** '2019-02-17' **and** startlocation = 'ottawa' **order by** price **asc**;

This query helps with searching a trip in a time period by start and end location. Final result was sorted by price to allow users to choose the most economical choice. This query selects trips starting at Ottawa and between Feb 14 2019 to Feb 17 2019, then selects stops with name Montreal and inner join them on trip id. Output was sorted by price ascendingly.

**select** overallrating **from** drivers **where** username **in** (**select** v.owner **from** vehicles v **group by** v.owner **having** *count*(\*) > 4);

This query searches all drivers with more than 4 cars and outputs their rating. It has a subquery and a parent query. Subquery groups all vehicles with owner (driver) and computer count of vehicles in each group, then outputs a set of driving ids (driver that has at least one vehicle). Parent query iterates all driver and checks whether the driver is in the set, outputs the overall rating for each driver.

**select** tripid **from** trips **where** price >= **all** (**select** price **from** trips);

This query selects the most expensive trip. Subqueries selects all price, and parent query selects the one that larger or equal to all prices, which is the largest one.

Cool SQL features

**select** t2.tripid, t2.price, t2.price - *avg*(t2.price) **over** (**partition by** t2.numberofseatsavailable) **as** realtivePricee **from** trips t2 **where** t2.tripid **in** (**select** t.tripid **from** trips t **join** hasstops **on** t.tripid = hasstops.tripid **where** hasstops.stopname = 'montreal' **and** t.starttime **between** '2019-02-14' **and** '2019-02-17' **and** t.startlocation = 'ottawa');

This query uses WINDOW FUNCTION that computes relative price of searched trips. Relative price was determined by the difference between price and the average price of the group (grouped by number of seats in a vehicle), for the price may be affected by size of vehicle.

Complex Analytical Query

**select c**.cityname, *count*(books.uid) **from** books **inner join** trips t **on** books.tripid = t.tripid **inner join** hasstops **h on** t.tripid = **h**.tripid **inner join** stops **s on h**.cityid = **s**.cityid **and h**.stopname = **s**.stopname **inner join** cities **c on s**.cityid = **c**.cityid **group by c**.cityname **order by** *count*(books.uid);

This query computes the most popular, determined by count of passengers from a start location to a stop city. A inner join was done among trips, books, has-stops, stops, and cities. This multi way joined relation was then grouped by start location and stop city, and a count of bookings was done for each group to generate a popularity score for raking.