QUERING THE DATABASE.

The following queries retrieve data that may be useful for backend developers. For example, the query needed to perform the most important search operation of any rental systems, i.e., allowing a traveler/user to search for all properties available in a certain city for certain dates.

```
/* Return all rentals available for a specific period. Provide staring
date and ending date of the reservation.
Manually modify the dates in the WHERE clause.*/

SELECT rentals.rental_id, rentals.title, rentals.price FROM
(SELECT rentals.rental_id, rentals.title, rentals.price,
count(rentals.rental_id) FROM rentals.rentals AS rentals
JOIN rentals.reservations AS reservations
ON rentals.rental_id = reservations.rental_id
WHERE '2027-06-08' NOT BETWEEN reservations.reservation_starts AND
reservations.reservation_ends
AND '2027-06-25' NOT BETWEEN reservations.reservation_starts AND
reservations.reservation_ends
GROUP BY rentals.rental_id
ORDER BY rentals.rental_id ASC) AS rentals
JOIN
(SELECT rentals.rental_id, count(rentals.rental_id) FROM
rentals.rentals AS rentals
JOIN rentals.reservations AS reservations
ON rentals.rental_id = reservations.rental_id
GROUP BY rentals.rental_id
ORDER BY rentals.rental_id ASC) AS rentals_complete
ON rentals.rental_id = rentals_complete.rental_id
WHERE rentals.count = rentals_complete.count
ORDER BY rentals.rental_id ASC;


/* Returns the average response time of a host. If a conversation was
never answered by the host, the query
considers the current time to calculate the average resposne time.
Therefore, if there is a message that
the host never answered, and the query is run in the future, the query
will yield a very large response time.
However, the query will work just fine on active databases*/
(WITH thread_starts AS (
SELECT message_id, user_traveler_id, user_host_id, sender, sent_at,
COALESCE(LAG((user_traveler_id, user_host_id, sender))
OVER ordered != (user_traveler_id, user_host_id, sender), true) AS
thread_start
FROM inbox.messages
WHERE user_host_id = 147
WINDOW ordered AS (PARTITION BY user_traveler_id, user_host_id ORDER
BY sent_at
)),
```

```
thread_responses AS(
SELECT user_traveler_id, user_host_id, sender, sent_at,
LEAD(sent_at) OVER ordered AS responded_at,
COUNT(*) OVER unordered AS threads
FROM thread_starts
WHERE thread_start
WINDOW ordered AS (PARTITION BY user_traveler_id, user_host_id ORDER
BY sent_at),
unordered AS (PARTITION BY user_traveler_id, user_host_id))

SELECT
AVG(COALESCE(responded_at, CURRENT_TIMESTAMP) - sent_at) AS
avg_response_time
FROM thread_responses
WHERE sender != user_host_id AND (responded_at IS NOT NULL OR threads
= 1)
GROUP BY user_host_id);

/* Return all rentals available for a specific period and a specific
city. Provide staring date and ending date of the reservation
and a city name. Manually modify the dates and cities in both WHERE
clause.*/
SELECT rentals.rental_id, rentals.title, rentals.price,
cities.city_name FROM
(SELECT rentals.rental_id, rentals.title, rentals.price,
rentals.address_id, count(rentals.rental_id) FROM rentals.rentals AS
rentals
JOIN rentals.reservations AS reservations
ON rentals.rental_id = reservations.rental_id
WHERE '2027-06-08' NOT BETWEEN reservations.reservation_starts AND
reservations.reservation_ends
AND '2027-06-25' NOT BETWEEN reservations.reservation_starts AND
reservations.reservation_ends
GROUP BY rentals.rental_id
ORDER BY rentals.rental_id ASC) AS rentals
JOIN
(SELECT rentals.rental_id, count(rentals.rental_id) FROM
rentals.rentals AS rentals
JOIN rentals.reservations AS reservations
ON rentals.rental_id = reservations.rental_id
GROUP BY rentals.rental_id
ORDER BY rentals.rental_id ASC) AS rentals_complete
ON rentals.rental_id = rentals_complete.rental_id
JOIN shared.addresses AS addresses
ON rentals.address_id = addresses.address_id
JOIN shared.cities AS cities
ON addresses.city_id = cities.city_id
WHERE rentals.count = rentals_complete.count
AND cities.city_name = 'Oslo'
ORDER BY rentals.rental_id ASC;

/* Calculate Average Rating for any Rental.*/
```

```
SELECT avg(rentals.reviews.rating)
FROM rentals.reviews
WHERE rentals.reviews.rental_id = 5;

/* Get all reservations for dates greater than the current date and
some extra INFO*/
SELECT reservations.reservation_id, reservations.rental_id,
reservations.reservation_starts,
reservations.reservation_ends, reservations.reservation_duration,
rentals.price,
reservations.applied_fees_and_discounts, reservations.daily_price,
reservations.total_price
FROM rentals.reservations AS reservations
JOIN rentals.rentals AS rentals
ON reservations.rental_id = rentals.rental_id
WHERE reservations.rental_id = 14 AND reservation_starts >
CURRENT_DATE;


/* Get all reservations that have custom_pricing, i.e., that the host
selected special prices for certain days*/
SELECT reservations.reservation_id, reservations.rental_id,
reservations.reservation_starts,
reservations.reservation_ends, reservations.reservation_duration,
rentals.price,
reservations.applied_fees_and_discounts,
custom_pricing.discounted_price, reservations.daily_price,
reservations.total_price
FROM rentals.reservations AS reservations
JOIN rentals.custom_pricing AS custom_pricing
ON reservations.rental_id = custom_pricing.rental_id
JOIN rentals.rentals AS rentals
ON rentals.rental_id = reservations.rental_id
WHERE reservations.rental_id = 14 AND
custom_pricing.scheduled_date BETWEEN reservations.reservation_starts
AND reservations.reservation_ends;
```