**CISC 7512X Midterm Exam**

For the below questions, use the following schema definition.

```
users(user_id, name, email, created_at)
properties(property_id, owner_id, name, city, price_per_night)
bookings(booking_id, property_id, user_id, start_date, end_date, total_price)
reviews(review_id, booking_id, rating, comment)
```

This is a schema for an online Bed & Breakfast.

1. Retrieve all columns from the Users table:
   (a) SELECT * FROM Properties;
   (b) SELECT * FROM Users;
   (c) SELECT name FROM Users;
   (d) SELECT user_id, email FROM Properties;
   (e) Other:
2. Get the names of all properties in 'New York':
   (a) SELECT name FROM Properties WHERE city = 'New York';
   (b) SELECT * FROM Users WHERE city = 'New York';
   (c) SELECT city FROM Properties;
   (d) SELECT name FROM Bookings WHERE city = 'New York';
   (e) Other:
3. Insert a new booking:
   (a) INSERT INTO Bookings (property_id, user_id, start_date, end_date, total_price) VALUES (101, 10, '2026-03-20', '2026-03-22', 240);
   (b) UPDATE Bookings SET total_price=240 WHERE booking_id=101;
   (c) SELECT * FROM Bookings;
   (d) DELETE FROM Bookings WHERE booking_id=101;
   (e) Other:
4. Find all reviews with rating 5:
   (a) SELECT * FROM Reviews WHERE rating = 5;
   (b) SELECT * FROM Bookings WHERE rating = 5;
   (c) SELECT review_id FROM Reviews;
   (d) SELECT * FROM Properties WHERE rating = 5;
   (e) Other:
5. Get emails of all users who booked property 10:
   (a) SELECT email FROM Users JOIN Bookings ON Users.user_id = Bookings.user_id WHERE property_id = 10;

(b) SELECT email FROM Users WHERE user_id = 10;

(c) SELECT email FROM Properties WHERE property_id = 10;

(d) SELECT email FROM Bookings;

(e) Other:

6. Count bookings per property:

(a) SELECT property_id, COUNT(*) FROM Bookings GROUP BY property_id;

(b) SELECT property_id, SUM(total_price) FROM Bookings;

(c) SELECT COUNT(*) FROM Properties;

(d) SELECT booking_id, COUNT(*) FROM Bookings;

(e) Other:

7. List properties and their owner names?

(a) SELECT Properties.name, Users.name FROM Properties JOIN Users ON Properties.owner_id = Users.user_id;

(b) SELECT Properties.name, Users.name FROM Bookings;

(c) SELECT Properties.name FROM Properties;

(d) SELECT name FROM Users;

(e) Other:

8. Calculate average rating per property:

(a) SELECT Properties.property_id, AVG(Reviews.rating) FROM Properties JOIN Bookings ON Properties.property_id = Bookings.property_id JOIN Reviews ON Bookings.booking_id = Reviews.booking_id GROUP BY Properties.property_id;

(b) SELECT AVG(rating) FROM Reviews;

(c) SELECT property_id, rating FROM Reviews;

(d) SELECT property_id, SUM(rating) FROM Reviews;

(e) Other:

9. Find users who booked more than 2 properties:

(a) SELECT user_id FROM Bookings GROUP BY user_id HAVING COUNT(*) > 2;

(b) SELECT user_id FROM Users WHERE COUNT(bookings) > 2;

(c) SELECT * FROM Bookings WHERE COUNT(*) > 2;

(d) SELECT user_id FROM Bookings WHERE COUNT(*) > 2;

(e) Other:

10. Get all properties that were never booked:

(a) SELECT * FROM Properties WHERE property_id NOT IN (SELECT property_id FROM Bookings);

(b) SELECT * FROM Properties JOIN Bookings;

(c) SELECT * FROM Properties WHERE property_id IN Bookings;

(d) SELECT * FROM Bookings WHERE property_id NOT IN Properties;

(e) Other:

11. Rank properties by total revenue:

(a) SELECT property_id, SUM(total_price) AS revenue FROM Bookings GROUP BY property_id ORDER BY revenue DESC;

(b) SELECT distinct property_id, SUM(total_price) OVER (PARTITION BY property_id) AS total_revenue, RANK() OVER (ORDER BY SUM(total_price) OVER (PARTITION BY property_id) DESC) AS revenue_rank FROM Bookings;

(c) SELECT property_id, total_price FROM Bookings ORDER BY total_price DESC;

(d) SELECT property_id, total_price, ROW_NUMBER() OVER (ORDER BY total_price) FROM Bookings;

(e) Other:

12. What fraction of properties are in 'New York':

(a) SELECT COUNT(*) / (SELECT COUNT(*) FROM Properties) FROM Properties WHERE city = 'New York';

(b) SELECT AVG(CASE WHEN city = 'New York' THEN 1 ELSE 0 END) AS fraction_nyc FROM Properties;

(c) SELECT SUM(price_per_night) / COUNT(*) FROM Properties WHERE city = 'New York';

(d) SELECT COUNT(city) / COUNT(*) FROM Properties;

(e) Other:

13. Find top 2 properties by revenue per city:

(a) WITH STATS AS (SELECT property_id, city, SUM(total_price) AS revenue, RANK() OVER (PARTITION BY city ORDER BY SUM(total_price) DESC) AS city_rank FROM Bookings JOIN Properties USING(property_id) GROUP BY property_id, city) SELECT * FROM STATS WHERE city_rank <= 2;

(b) SELECT property_id, city FROM Properties ORDER BY total_price DESC LIMIT 2;

(c) SELECT property_id, SUM(total_price) FROM Bookings GROUP BY city;

(d) SELECT * FROM Properties WHERE revenue > 0;

(e) Other:

14. Calculate each user's total spent and rank them:

(a) SELECT user_id, SUM(total_price) AS total_spent
FROM Bookings
GROUP BY user_id ORDER BY total_spent DESC;

(b) SELECT user_id,
SUM(total_price) OVER (PARTITION BY user_id) AS total_spent,
RANK() OVER (ORDER BY SUM(total_price) OVER (PARTITION BY user_id) DESC)
AS rank FROM Bookings;

(c) SELECT user_id, SUM(total_price) tot, RANK() over (partition by user_id order by SUM(total_price) DESC) rank FROM Bookings GROUP BY user_id

(d) SELECT user_id, COUNT(*) FROM Bookings GROUP BY user_id;

(e) Other:

15. List users who booked properties in multiple cities?

(a) SELECT user_id FROM Bookings JOIN Properties USING(property_id)
GROUP BY user_id HAVING COUNT(DISTINCT city) > 1;

(b) SELECT DISTINCT user_id FROM Bookings WHERE city IN Properties;

(c) SELECT user_id FROM Users WHERE COUNT(DISTINCT city) > 1;

(d) SELECT user_id FROM Bookings WHERE city > 1;

(e) Other:

16. Find the owner with highest total revenue:

(a) SELECT owner_id, SUM(total_price) FROM Properties JOIN Bookings USING(property_id)
GROUP BY owner_id ORDER BY SUM(total_price) DESC LIMIT 1;

(b) SELECT owner_id, COUNT(*) FROM Bookings JOIN Properties;

(c) SELECT owner_id, MAX(total_price) FROM Properties;

(d) SELECT user_id, SUM(total_price) FROM Bookings;

(e) Other:

17. Compute cumulative revenue over time per property:

(a) SELECT property_id, start_date,
SUM(total_price) OVER (PARTITION BY property_id ORDER BY start_date) AS cum_revenue
FROM Bookings;

(b) SELECT property_id, SUM(total_price)
FROM Bookings GROUP BY property_id;

(c) SELECT property_id, total_price FROM Bookings;

(d) SELECT property_id, AVG(total_price) FROM Bookings;

(e) Other:

18. Find properties with higher than average revenue per city:

(a) SELECT property_id, city FROM Properties;

(b) WITH STATS AS (
SELECT property_id, city, SUM(total_price) AS revenue,
AVG(SUM(total_price)) OVER (PARTITION BY city) avgrev
FROM Bookings JOIN Properties USING(property_id)
GROUP BY property_id, city)
SELECT * FROM STATS WHERE revenue > avgrev;

(c) SELECT property_id FROM Bookings WHERE total_price > 1000;

(d) SELECT * FROM Properties WHERE revenue > 1000;

(e) Other:

19. Sort users by average rating they gave:

    (a) SELECT user_id, AVG(rating) AS avg_rating
    FROM Bookings JOIN Reviews USING(booking_id)
    GROUP BY user_id ORDER BY avg_rating DESC;

    (b) SELECT user_id, rating FROM Reviews;

    (c) SELECT user_id, SUM(rating) FROM Reviews;

    (d) SELECT user_id, COUNT(*) FROM Reviews;

    (e) Other:

20. Find the top reviewer in terms of number of reviews:

    (a) SELECT user_id, COUNT(*) AS num_reviews
    FROM Bookings JOIN Reviews USING(booking_id)
    GROUP BY user_id ORDER BY num_reviews DESC LIMIT 1;

    (b) SELECT user_id, rating FROM Reviews;

    (c) SELECT user_id, AVG(rating) FROM Reviews;

    (d) SELECT * FROM Reviews ORDER BY rating DESC;

    (e) Other: