CISC 7510X 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. Query to 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. Query to find the name and email of all users?
   (a) SELECT name, email FROM Users;
   (b) SELECT * FROM Users;
   (c) SELECT user_id, email FROM Users;
   (d) SELECT name, email FROM Properties;
   (e) Other:
3. Query to list all properties in 'New York'?
   (a) SELECT * FROM Properties WHERE city = 'New York';
   (b) SELECT * FROM Users WHERE city = 'New York';
   (c) SELECT * FROM Bookings WHERE city = 'New York';
   (d) SELECT * FROM Reviews WHERE city = 'New York';
   (e) Other:
4. Count how many bookings each user has made?
   (a) SELECT user_id, COUNT(*) FROM Bookings GROUP BY user_id;
   (b) SELECT COUNT(user_id) FROM Bookings;
   (c) SELECT user_id FROM Bookings WHERE COUNT(*) > 1;
   (d) SELECT user_id, SUM(total_price) FROM Bookings;
   (e) Other:
5. Retrieve the property name and the name of the user who booked it?
   (a) SELECT Properties.name, Users.name FROM Properties INNER JOIN Bookings ON Properties.property_id = Bookings.property_id INNER JOIN Users ON Bookings.user_id = Users.user_id;

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

(c) SELECT Properties.name, Users.name FROM Users INNER JOIN Bookings ON Users.user_id = Bookings.user_id;

(d) SELECT Properties.name, Users.name FROM Reviews INNER JOIN Users ON Reviews.booking_id = Users.user_id;

(e) Other:

6. Statement to add a new property:

(a) INSERT INTO Properties (owner_id, name, city, price_per_night) VALUES (10, 'Cozy Loft', 'Boston', 120);

(b) UPDATE Properties SET city = 'Boston' WHERE property_id = 10;

(c) SELECT * FROM Properties;

(d) DELETE FROM Properties WHERE property_id = 10;

(e) Other:

7. Find all users who have never made a booking?

(a) SELECT * FROM Users JOIN Bookings;

(b) SELECT * FROM Users WHERE user_id IN Bookings;

(c) SELECT * FROM Bookings WHERE user_id NOT IN Users;

(d) SELECT * FROM Users WHERE user_id NOT IN (SELECT user_id FROM Bookings);

(e) Other:

8. Calculate average rating per property?

(a) SELECT AVG(rating) FROM Reviews;

(b) 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;

(c) SELECT property_id, rating FROM Reviews;

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

(e) Other:

9. Find total revenue per property?

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

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

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

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

(e) Other:

10. Find users who booked more than 3 properties?

(a) SELECT user_id FROM Bookings WHERE COUNT(*) > 3;

(b) SELECT * FROM Users WHERE COUNT(bookings) > 3;

(c) SELECT user_id FROM Bookings WHERE COUNT(bookings) > 3;

(d) SELECT user_id FROM Bookings GROUP BY user_id HAVING COUNT(*) > 3;

(e) Other:

11. Find users who booked less than 3 properties?

(a) SELECT user_id FROM Bookings WHERE COUNT(*) < 3;

(b) SELECT * FROM Users LEFT OUTER JOIN Bookings on Users.user_id=Bookings.user_id group by Users.user_id having COUNT(Bookings.booking_id) < 3;

(c) SELECT user_id FROM Bookings WHERE COUNT(bookings) < 3;

(d) SELECT user_id FROM Bookings GROUP BY user_id HAVING COUNT(*) < 3;

12. Find the highest booking price for each property.

(a) SELECT MAX(total_price) FROM Bookings;

(b) SELECT property_id, total_price FROM Bookings WHERE total_price = MAX(total_price);

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

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

13. Find properties with no reviews:

(a) SELECT * FROM Properties WHERE property_id IN Reviews;

(b) SELECT * FROM Reviews WHERE property_id IS NULL;

(c) SELECT * FROM Properties INNER JOIN Reviews;

(d) SELECT * FROM Properties WHERE property_id NOT IN (SELECT property_id FROM Bookings INNER JOIN Reviews ON Bookings.booking_id = Reviews.booking_id);

14. Find the owner who earned the most revenue?

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

(b) SELECT owner_id, SUM(total_price) FROM Properties INNER JOIN Bookings ON Properties.property_id = Bookings.property_id GROUP BY owner_id ORDER BY SUM(total_price) DESC LIMIT 1;

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

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

15. Find users who booked properties in multiple cities?

(a) SELECT user_id FROM Bookings GROUP BY user_id HAVING COUNT(city) > 1;

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

(c) SELECT user_id FROM Bookings JOIN Properties ON Bookings.property_id = Properties.property_id GROUP BY user_id HAVING COUNT(DISTINCT city) > 1;

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

16. Find users who booked properties in multiple cities?

(a) SELECT user_id FROM Bookings GROUP BY user_id HAVING COUNT(city) > 1;

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

(c) SELECT user_id FROM Bookings JOIN Properties ON Bookings.property_id = Properties.property_id GROUP BY user_id HAVING COUNT(DISTINCT city) > 1;

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

17. What fraction of properties are in NYC?

(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;

18. Rank all 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 property_id, total_price, RANK() OVER (ORDER BY SUM(total_price)) AS revenue_rank FROM Bookings;

(c) 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;

(d) SELECT property_id, total_price FROM Bookings WHERE RANK() OVER (ORDER BY total_price DESC) = 1;

19. In general, on *limited memory system*, no indexes, and huge tables, what join type would perform best?

(a) merge join.

(b) hash join.

(c) indexed lookup join.

(d) inner loop join.

20. What is the most appropriate join type for this query:
SELECT * FROM Bookings INNER JOIN Properties ON Bookings.property_id = Properties.property_id

(a) inner loop join

(b) bitmap tree loop join

(c) sort-merge join

(d) hashmap join