SR	Questions	Option 1	Option 2	Option 3	Option 4	Ans
1	Which of the following is NOT a benefit of using transactions?	Data integrity	High availability	Data consistency	Data durablity	B
2	A transaction that violates the consistency property is considered to be:	Serializable	Inconsistent	Isolate	Error	B
3	Can you change the parameter values of a cursor after it has been declared and opened?	Yes, parameter values can be modified at any time.	No, parameter values are fixed once the cursor is declared and opened.	Parameter values can only be changed during cursor declaration.	Cursors cannot have parameter values.	B
4	Can you declare a cursor without specifying the SELECT statement immediately?	No, a SELECT statement must always be specified.	Yes, a SELECT statement can be added later in the code.	Cursors cannot be declared in PL/SQL.	Cursors are automatically generated in PL/SQL.	A
5	Can you declare multiple cursors with the same name but different parameters in the same PL/SQL block?	Yes, as long as the cursor names are unique.	No, cursor names must be unique regardless of the parameters.	Multiple cursors are not allowed in the same PL/SQL block.	Cursors with parameters cannot have the same name.	B
6	Can you declare multiple cursors within the same PL/SQL block? If so, how do you differentiate them?	No, only one cursor is allowed per block.	Yes, multiple cursors can be declared, and they are differentiated by their data types.	Yes, multiple cursors can be declared, and they are differentiated by their names.	Multiple cursors cannot be used in PL/SQL.	C
7	Can you fetch data from a cursor into individual variables or into a record type? Explain.	Data can only be fetched into individual variables.	Data can only be fetched into a record type.	Data can be fetched into both individual variables and a record type.	Data cannot be fetched from a cursor.	C

8	Can you nest a Cursor FOR Loop inside another Cursor FOR Loop? If so, why might you do so?	No, nesting Cursor FOR Loops is not allowed.	Yes, you can nest Cursor FOR Loops to perform complex data processing and handle related data hierarchies.	Cursor FOR Loops can only be used individually, not nested.	Nesting Cursor FOR Loops results in performance issues.	B

9	Can you use a Cursor FOR Loop to update or delete records in a database table? Explain.	No, Cursor FOR Loops are read- only.	Yes, Cursor FOR Loops can update or delete records using the UPDATE and DELETE statements.	Cursor FOR Loops can only insert records, not update or delete them.	Cursor FOR Loops can only be used for reporting purposes.	B
10	Describe the differences between an implicit cursor and an explicit cursor in PL/SQL.	Implicit cursors are used for data modeling, while explicit cursors are used for data manipulation.	Implicit cursors are automatically created for DML statements, while explicit cursors are user-defined.	Implicit cursors are used for database connections, while explicit cursors are used for loop control.	Implicit cursors are used for hardware design, while explicit cursors are used for web development.	B
11	Describe the purpose of PL/SQL collections, and provide examples of their types.	PL/SQL collections are used for defining variables.	PL/SQL collections are used for database connections.	PL/SQL collections are used for storing multiple values of the same data type.	PL/SQL collections are used for creating triggers.	C
12	Explain how cursor parameters can be used to create dynamic cursors.	Cursor parameters have no role in creating dynamic cursors.	By allowing parameterization of the WHERE clause in the cursor's SELECT statement, you can create dynamic cursors that retrieve specific data based on different criteria.	Cursor parameters can only be used with static cursors.	Cursor parameters can be used to create triggers.	B

13	Explain the concept of triggers in a database context. How are they used in PL/SQL?	Triggers are used for creating web applications.	Triggers are used for hardware design.	Triggers are used for automatically executing PL/SQL code in response to database events.	Triggers are used for data modeling.	C
14	Explain the difference between declaring a cursor and opening a cursor.	Declaring a cursor retrieves data; opening a cursor defines its structure.	Declaring a cursor defines its structure; opening a cursor retrieves data.	Declaring a cursor and opening a cursor are the same.	Declaring a cursor is not a PL/SQL concept.	B
15	Explain the importance of transactions in PL/SQL and how they are managed.	Transactions are used for web development.	Transactions are used for data modeling.	Transactions ensure data consistency and are managed using COMMIT and ROLLBACK statements.	Transactions are not supported in PL/SQL.	C

16	Explain the purpose of a PL/SQL package and its components.	PL/SQL packages are used for web development.	PL/SQL packages are used for encapsulating procedures and functions.	PL/SQL packages are used for data modeling.	PL/SQL packages are used for hardware design.	B
17	How can you pass parameters to a PL/SQL procedure or function?	Parameters are passed using the CALL statement.	Parameters are not supported in PL/SQL.	Parameters are passed as input and output variables.	Parameters are passed using the DECLARE statement.	C
18	How can you resolve a deadlock in a database system?	By terminating one of the transactions involved in the deadlock.	By rolling back all transactions involved in the deadlock.	By increasing the isolation level.	Deadlocks cannot be resolved.	A
19	How do you create and manipulate PL/SQL associative arrays (index-by tables)?	Associative arrays are created using the ARRAY keyword.	Associative arrays are created using the INDEX keyword.	Associative arrays are not supported in PL/SQL.	Associative arrays are created using the TYPE keyword.	D
20	How do you declare a cursor, and what are the required components?	Cursors are automatically declared in PL/SQL.	Cursors are declared using the DECLARE CURSOR statement and require a SELECT statement.	Cursors are declared using the DECLARE keyword.	Cursors are declared using the OPEN statement.	B

21	How do you declare a variable in PL/SQL, and what are the data types supported for variables?	Variables are declared using the DECLARE keyword, and PL/SQL supports only one data type.	Variables are declared using the VAR keyword, and PL/SQL supports multiple data types.	Variables are declared using the VARIABLE keyword, and PL/SQL supports multiple data types.	Variables are not supported in PL/SQL.	B
22	How do you define and use PL/SQL records and record types?	Records are used for creating tables in PL/SQL.	Records are defined using the DECLARE RECORD statement.	Records are used to hold data in a structured format.	Records are not supported in PL/SQL.	C
23	How do you ensure that you've fetched all available data from a cursor?	By using the CLOSE statement.	By using the OPEN statement.	By checking the cursor attribute %NOTFOUND.	Cursors automatically fetch all available data.	C
24	How do you handle database connections and transactions in PL/SQL?	Database connections and transactions are automatically managed by the PL/SQL engine.	Database connections and transactions are not supported in PL/SQL.	Database connections are established using the CONNECT statement, and transactions are managed using COMMIT and ROLLBACK statements.	Database connections are established using the DECLARE statement.	C

Given this Restaurant schema Answer the following Questions

Tables:

a. Customers:

- `customer_id` (Primary Key)
- `first_name`
- `last_name`
- `email`
- `phone_number`
- `address`
- ...

b. Employees:

- `employee_id` (Primary Key)
- `first_name`
- `last_name`
- `email`
- `phone_number`
- `position`
- `hire_date`
- ...

c. Menu Items:

- `item_id` (Primary Key)
- `name`
- `description`
- `price`
- `category`
- ...

d. Orders:

- `order_id` (Primary Key)
- `customer_id` (Foreign Key to Customers)
- `order_date`
- `total_amount`
- ...

e. Order Items:

- `order_item_id` (Primary Key)
- `order_id` (Foreign Key to Orders)
- `item_id` (Foreign Key to Menu Items)
- `quantity`
- `subtotal`
- ...

f. Reservations:

- `reservation_id` (Primary Key)
- `customer_id` (Foreign Key to Customers)
- `reservation_date`
- `table_number`
- `party_size`

- ...

g. Tables:
- `table_number` (Primary Key)
- `seating_capacity`
- `description`
- `status` (e.g., available, occupied, reserved)
- ...

h. Reviews:
- `review_id` (Primary Key)
- `customer_id` (Foreign Key to Customers)
- `rating`
- `comments`
- `review_date`
- ...

i. Payments:
- `payment_id` (Primary Key)
- `order_id` (Foreign Key to Orders)
- `payment_date`
- `payment_amount`
- `payment_method`
- ...

Question 1: In the restaurant database schema, which table would likely store information about the dishes and drinks available for customers to order?

A. Customers
B. Employees
C. Menu Items
D. Orders

Question 2: What is the primary key for the "Employees" table in the restaurant database schema?

A. employee_id
B. order_id
C. customer_id
D. table_number

Question 3: If you want to find the total number of reservations made by a specific customer with the first name "John," which SQL statement would you use?

A. SELECT COUNT(*) FROM Reservations WHERE first_name = 'John';
B. SELECT COUNT(*) FROM Reservations WHERE customer_id = 'John';
C. SELECT COUNT(*) FROM Reservations WHERE reservation_date = 'John';
D. SELECT COUNT(*) FROM Reservations WHERE party_size = 'John';

Question 4: What is the foreign key in the "Order Items" table in the restaurant database schema?

A. order_id
B. order_item_id
C. customer_id
D. item_id

Question 5: Which SQL statement would you use to find the average rating of all reviews in the "Reviews" table?

A. SELECT AVG(rating) FROM Reviews;
B. SELECT AVG(rating) FROM Reviews WHERE rating IS NOT NULL;
C. SELECT AVERAGE(rating) FROM Reviews;
D. SELECT AVG(rating) FROM Reviews GROUP BY rating;

Question 6: What data type is typically used for the "price" field in the "Menu Items" table to represent monetary values?

A. INT
B. FLOAT
C. VARCHAR
D. DECIMAL

Question 7: Which SQL clause is used to filter rows in a query result, based on a specified condition?

A. GROUP BY
B. HAVING
C. WHERE
D. ORDER BY

Question 8: If you want to retrieve all the menu items with a price less than $10, which SQL statement would you use?

A. SELECT * FROM "Menu Items" WHERE price < 10;
B. SELECT * FROM "Menu Items" WHERE price > 10;
C. SELECT * FROM "Menu Items" HAVING price < 10;
D. SELECT * FROM "Menu Items" ORDER BY price ASC;

Question 9: What SQL clause is used to sort the result set in ascending or descending order?

A. WHERE
B. SORT
C. ORDER BY
D. GROUP BY

Question 10: In the "Tables" table of the restaurant schema, what field indicates whether a table is currently available for seating?

A. seating_capacity
B. description
C. status
D. table_number

Question 11: Which SQL statement would you use to find the highest total amount spent on an order?

A. SELECT MAX(total_amount) FROM Orders;
B. SELECT MIN(total_amount) FROM Orders;
C. SELECT SUM(total_amount) FROM Orders;
D. SELECT AVG(total_amount) FROM Orders;

Question 12: In the "Payments" table, which field would typically store the payment method used for a particular order?

A. payment_id
B. order_id
C. payment_date
D. payment_method

Question 13: What is the purpose of the SQL GROUP BY clause?

A. To filter rows based on a condition.
B. To join two or more tables.
C. To aggregate data and perform calculations on groups of rows.
D. To sort rows in a result set.

Question 14: Which SQL statement is used to add a new customer to the "Customers" table?

A. INSERT INTO Customers (first_name, last_name) VALUES ('John', 'Doe');
B. UPDATE Customers SET first_name = 'John', last_name = 'Doe' WHERE customer_id = 1;
C. DELETE FROM Customers WHERE first_name = 'John' AND last_name = 'Doe';
D. SELECT * FROM Customers WHERE first_name = 'John' AND last_name = 'Doe';

Question 15: In the "Reservations" table, what does the "party_size" field represent?

A. The reservation ID
B. The number of people in the reservation party
C. The table number
D. The reservation date and time

Question 16: What SQL statement is used to delete a specific order with order_id = 12345 from the "Orders" table?

A. DELETE FROM Orders WHERE order_id = 12345;
B. UPDATE Orders SET order_status = 'Canceled' WHERE order_id = 12345;
C. SELECT * FROM Orders WHERE order_id = 12345;
D. INSERT INTO Orders (order_id) VALUES (12345);

Question 17: In the "Customers" table, what is the primary key for uniquely identifying each customer?

A. order_id
B. customer_id
C. employee_id
D. menu_item_id

Question 18: Which SQL clause is used to combine rows from two or more tables based on a related column between them?

A. WHERE
B. JOIN
C. GROUP BY
D. HAVING

Question 19: If you want to find the names and email addresses of customers who have made a reservation, which SQL statement would you use?

A. SELECT first_name, email FROM Customers;
B. SELECT first_name, email FROM Reservations;
C. SELECT first_name, email FROM Customers WHERE customer_id IN (SELECT customer_id FROM Reservations);
D. SELECT first_name, email FROM Reservations WHERE customer_id IN (SELECT customer_id FROM Customers);

Question 20: In the "Menu Items" table, which SQL constraint should ensure that the "name" field contains unique values for each menu item?

A. PRIMARY KEY
B. FOREIGN KEY
C. UNIQUE
D. CHECK

Answers:

1. C. Menu Items
2. A. employee_id
3. A. SELECT COUNT(*) FROM Reservations WHERE first_name = 'John';
4. A. order_id
5. A. SELECT AVG(rating) FROM Reviews;
6. D. DECIMAL
7. C. WHERE
8. A. SELECT * FROM "Menu Items" WHERE price < 10;
9. C. ORDER BY
10. C. status
11. A. SELECT MAX(total_amount) FROM Orders;
12. D. payment_method
13. C. To aggregate data and perform calculations on groups of rows.
14. A. INSERT INTO Customers (first_name, last_name) VALUES ('John', 'Doe');
15. B. The number of people in the reservation party
16. A. DELETE FROM Orders WHERE order_id = 12345;
17. B. customer_id
18. B. JOIN
19. C. SELECT first_name, email FROM Customers WHERE customer_id IN (SELECT customer_id FROM Reservations);
20. C. UNIQUE

Question 1: You want to retrieve the names of customers who have placed orders but have not made any reservations. Which SQL query should you use?

A. SELECT c.first_name, c.last_name FROM Customers c WHERE c.customer_id IN (SELECT o.customer_id FROM Orders o) AND c.customer_id NOT IN (SELECT r.customer_id FROM Reservations r);

B. SELECT c.first_name, c.last_name FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id WHERE c.customer_id NOT IN (SELECT r.customer_id FROM Reservations r);

C. SELECT c.first_name, c.last_name FROM Customers c WHERE c.customer_id NOT IN (SELECT r.customer_id FROM Reservations r) GROUP BY c.customer_id HAVING COUNT(o.customer_id) > 0;

D. SELECT c.first_name, c.last_name FROM Customers c LEFT JOIN Reservations r ON c.customer_id = r.customer_id WHERE r.customer_id IS NULL;

Question 2: In the restaurant database schema, which SQL statement would you use to find the customer who has spent the most on orders?

A. SELECT MAX(total_amount) FROM Orders;

B. SELECT c.first_name, c.last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
HAVING MAX(o.total_amount);

C. SELECT c.first_name, c.last_name
FROM Customers c
WHERE c.customer_id = (SELECT customer_id FROM Orders WHERE total_amount = (SELECT MAX(total_amount) FROM Orders));

D. SELECT c.first_name, c.last_name
FROM Customers c
WHERE c.customer_id = (SELECT customer_id FROM Orders GROUP BY customer_id HAVING MAX(total_amount));

Question 3: What is the result of the following SQL query?

```sql
SELECT COUNT(*)
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
WHERE o.order_date > (SELECT MAX(reservation_date) FROM Reservations WHERE customer_id = o.customer_id);
```

A. It counts the number of orders placed after the last reservation date for each customer.

B. It counts the number of orders placed by each customer.

C. It counts the number of customers who have placed orders after making reservations.

D. It returns an error because the subquery is not properly correlated.

Question 4: To find the average number of orders placed by customers, which SQL statement should you use?

A. SELECT AVG(COUNT(order_id)) FROM Orders;

B. SELECT AVG(order_count) FROM (SELECT customer_id, COUNT(order_id) as order_count FROM Orders GROUP BY customer_id) AS subquery;

C. SELECT AVG(COUNT(order_id)) FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id GROUP BY c.customer_id;

D. SELECT AVG(order_count) FROM (SELECT customer_id, COUNT(order_id) as order_count FROM Orders GROUP BY customer_id) subquery;

Question 5: You want to find the names of employees who have processed at least one payment and also placed an order. Which SQL statement would you use?

A.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  JOIN Payments p ON e.employee_id = p.employee_id
  WHERE e.employee_id IN (SELECT employee_id FROM Orders);
```

B.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  WHERE e.employee_id IN (SELECT employee_id FROM Orders)
  AND e.employee_id IN (SELECT employee_id FROM Payments);
```

C.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  JOIN Payments p ON e.employee_id = p.employee_id
  JOIN Orders o ON e.employee_id = o.employee_id;
```

D.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  WHERE e.employee_id IN (SELECT employee_id FROM Orders)
  UNION
  SELECT e.first_name, e.last_name
  FROM Employees e
  WHERE e.employee_id IN (SELECT employee_id FROM Payments);
```

Question 6: Which SQL statement would you use to find the names of customers who have not placed any orders and have not made any reservations?

A.
```
SELECT c.first_name, c.last_name
  FROM Customers c
  WHERE c.customer_id NOT IN (SELECT customer_id FROM Orders)
  AND c.customer_id NOT IN (SELECT customer_id FROM Reservations);
```

B. SELECT c.first_name, c.last_name
FROM Customers c
LEFT JOIN Orders o ON c.customer_id = o.customer_id
LEFT JOIN Reservations r ON c.customer_id = r.customer_id
WHERE o.customer_id IS NULL AND r.customer_id IS NULL;

C. SELECT c.first_name, c.last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Reservations r ON c.customer_id = r.customer_id
WHERE o.customer_id IS NULL AND r.customer_id IS NULL;

D. It is not possible to find such customers with a single SQL query.

Question 7: In the "Payments" table, which SQL constraint would ensure that the "payment_date" is not null?

A. PRIMARY KEY

B. FOREIGN KEY

C. NOT NULL

D. CHECK

Question 8: You want to find the menu items with the highest price in each category. Which SQL statement would you use?

A. SELECT MAX(price), category FROM "Menu Items" GROUP BY category;

B. SELECT category, MAX(price) FROM "Menu Items" GROUP BY category;

C. SELECT category, price FROM "Menu Items" WHERE price = MAX(price) GROUP BY category;

D. SELECT category, price FROM "Menu Items" WHERE price IN (SELECT MAX(price) FROM "Menu Items" GROUP BY category);

Question 9: What does the following SQL query do?

```sql
SELECT c.first_name, c.last_name
FROM Customers c
WHERE c.customer_id NOT IN (
    SELECT o.customer_id
    FROM Orders o
    WHERE o.order_date >= '2023-01-01'
);
```

A. It selects the names of customers who have placed orders on or after January 1, 2023.

B. It selects the names of customers who have never placed an order.

C. It selects the names of customers who have placed orders before January 1, 2023.

D. It selects the names of customers who have placed orders before or on January 1, 2023.

Question 10: In the "Reviews" table, you want to find the average rating given by customers for each menu item. Which SQL statement should you use?

A. SELECT AVG(rating), item_id FROM Reviews GROUP BY item_id;

B. SELECT AVG(rating), menu_item_id FROM Reviews GROUP BY menu_item_id;

C. SELECT AVG(rating), menu_item_id FROM Menu Items GROUP BY menu_item_id;

D. SELECT AVG(rating), item_id FROM Menu Items GROUP BY item_id;

Question 11: To find the total amount spent by each customer on their orders, which SQL statement should you use?

A. SELECT customer_id, SUM(total_amount) FROM Orders;

B. SELECT c.first_name, c.last_name, SUM(o.total_amount)
FROM Customers c
JOIN Orders o ON

c.customer_id = o.customer_id
GROUP BY c.customer_id;

C. SELECT SUM(total_amount) FROM Orders GROUP BY customer_id;

D. SELECT c.first_name, c.last_name, total_amount FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id;

Question 12: What is the result of the following SQL query?

```sql
SELECT c.first_name, c.last_name
FROM Customers c
WHERE c.customer_id IN (
    SELECT o.customer_id
    FROM Orders o
    WHERE o.total_amount > 50
);
```

A. It selects the names of customers who have placed orders with a total amount greater than 50.

B. It selects the names of customers who have never placed an order.

C. It selects the names of customers who have placed orders with a total amount less than or equal to 50.

D. It returns an error because the subquery is not properly correlated.

Question 13: To find the names of customers who have placed at least two orders, which SQL statement should you use?

A. SELECT c.first_name, c.last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
HAVING COUNT(o.order_id) >= 2;

B. SELECT c.first_name, c.last_name
FROM Customers c
WHERE c.customer_id IN (SELECT customer_id FROM Orders GROUP BY customer_id HAVING COUNT(order_id) >= 2);

C. SELECT c.first_name, c.last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
WHERE COUNT(o.order_id) >= 2;

D. SELECT c.first_name, c.last_name
FROM Customers c
WHERE c.customer_id IN (SELECT customer_id FROM Orders WHERE COUNT(order_id) >= 2);

Question 14: What does the following SQL query do?

```sql
SELECT m.name
FROM "Menu Items" m
WHERE m.price = (
    SELECT MAX(price)
    FROM "Menu Items"
    WHERE category = 'Appetizers'
);
```

A. It selects the names of the most expensive menu items in the "Appetizers" category.

B. It selects the names of all menu items in the "Appetizers" category.

C. It selects the names of the most expensive menu items in all categories.

D. It returns an error because the subquery is not properly correlated.

Question 15: You want to find the names of customers who have placed orders, made reservations, and left a review. Which SQL statement would you use?

A. SELECT c.first_name, c.last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id

```
  JOIN Reservations r ON c.customer_id = r.customer_id
  JOIN Reviews rv ON c.customer_id = rv.customer_id;
```

B.
```
SELECT c.first_name, c.last_name
  FROM Customers c
  WHERE c.customer_id IN (SELECT customer_id FROM Orders)
  AND c.customer_id IN (SELECT customer_id FROM Reservations)
  AND c.customer_id IN (SELECT customer_id FROM Reviews);
```

C.
```
SELECT c.first_name, c.last_name
  FROM Customers c
  JOIN Orders o ON c.customer_id = o.customer_id
  JOIN Reservations r ON c.customer_id = r.customer_id
  WHERE c.customer_id IN (SELECT customer_id FROM Reviews);
```

D. It is not possible to find such customers with a single SQL query.

Question 16: You want to find the total number of orders placed on tables with a seating capacity of 4 or more. Which SQL query should you use?

A. SELECT COUNT(*) FROM Orders o JOIN "Tables" t ON o.table_number = t.table_number WHERE t.seating_capacity >= 4;

B. SELECT COUNT(*) FROM Orders o WHERE o.table_number IN (SELECT table_number FROM "Tables" WHERE seating_capacity >= 4);

C. SELECT COUNT(*) FROM Orders o JOIN "Tables" t ON o.table_number = t.table_number WHERE t.seating_capacity <= 4;

D. SELECT COUNT(*) FROM Orders o WHERE o.table_number IN (SELECT table_number FROM "Tables" WHERE seating_capacity <= 4);

Question 17: To find the menu items that have never been reviewed, which SQL statement should you use?

A. SELECT name FROM "Menu Items" WHERE item_id NOT IN (SELECT item_id FROM Reviews);

B. SELECT name FROM "Menu Items" WHERE item_id IN (SELECT item_id FROM Reviews) HAVING COUNT(*) = 0;

C. SELECT name FROM "Menu Items" WHERE NOT EXISTS (SELECT * FROM Reviews WHERE Reviews.item_id = "Menu Items".item_id);

D. SELECT name FROM "Menu Items" LEFT JOIN Reviews ON "Menu Items".item_id = Reviews.item_id WHERE Reviews.item_id IS NULL;

Question 18: You want to find the names of customers who have placed orders on or after the date of their last reservation. Which SQL query should you use?

A.
```
SELECT c.first_name, c.last_name
  FROM Customers c
  JOIN Orders o ON c.customer_id = o.customer_id
```

```
  JOIN Reservations r ON c.customer_id = r.customer_id
  WHERE o.order_date >= r.reservation_date;
```

B.
```
SELECT c.first_name, c.last_name
  FROM Customers c
  WHERE c.customer_id IN (
    SELECT customer_id
    FROM Orders
    WHERE order_date >= (SELECT MAX(reservation_date) FROM Reservations WHERE customer_id =
c.customer_id)
  );
```

C.
```
SELECT c.first_name, c.last_name
  FROM Customers c
  JOIN Orders o ON c.customer_id = o.customer_id
  JOIN Reservations r ON c.customer_id = r.customer_id
  WHERE o.order_date >= (SELECT MAX(reservation_date) FROM Reservations WHERE customer_id =
c.customer_id);
```

D. It is not possible to achieve this with a single SQL query.

Question 19: You want to find the employees who have processed payments for orders with a total amount greater than $1000. Which SQL query should you use?

A.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  JOIN Payments p ON e.employee_id = p.employee_id
  WHERE p.order_id IN (SELECT order_id FROM Orders WHERE total_amount > 1000);
```

B.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  JOIN Payments p ON e.employee_id = p.employee_id
  WHERE p.order_id IN (SELECT order_id FROM Orders WHERE total_amount > 1000);
```

C.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  JOIN Orders o ON e.employee_id = o.employee_id
  JOIN Payments p ON o.order_id = p.order_id
  WHERE o.total_amount > 1000;
```

D.
```
SELECT e.first_name, e.last_name
  FROM Employees e
  WHERE e.employee_id IN (
    SELECT p.employee_id
    FROM Payments p
    WHERE p.order_id IN (SELECT order_id FROM Orders WHERE total_amount > 1000)
  );
```

Question 20: To find the number of customers who have both placed orders and made reservations, which SQL query should you use?

A. SELECT COUNT(DISTINCT c.customer_id)
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Reservations

r ON c.customer_id = r.customer_id;

B. SELECT COUNT(*) FROM Customers c WHERE c.customer_id IN (SELECT customer_id FROM Orders) AND c.customer_id IN (SELECT customer_id FROM Reservations);

C. SELECT COUNT(*) FROM Customers c WHERE c.customer_id IN (SELECT customer_id FROM Orders) INTERSECT SELECT customer_id FROM Reservations;

D. It is not possible to find such customers with a single SQL query.

Answers:

1. B. SELECT c.first_name, c.last_name FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id WHERE c.customer_id NOT IN (SELECT r.customer_id FROM Reservations r);
2. C. SELECT c.first_name, c.last_name FROM Customers c WHERE c.customer_id = (SELECT customer_id FROM Orders WHERE total_amount = (SELECT MAX(total_amount) FROM Orders));
3. A. It counts the number of orders placed after the last reservation date for each customer.
4. B. SELECT AVG(order_count) FROM (SELECT customer_id, COUNT(order_id) as order_count FROM Orders GROUP BY customer_id) AS subquery;
5. A. SELECT e.first_name, e.last_name FROM Employees e JOIN Payments p ON e.employee_id = p.employee_id WHERE e.employee_id IN (SELECT employee_id FROM Orders);
6. B. SELECT c.first_name, c.last_name FROM Customers c LEFT JOIN Orders o ON c.customer_id = o.customer_id LEFT JOIN Reservations r ON c.customer_id = r.customer_id WHERE o.customer_id IS NULL AND r.customer_id IS NULL;
7. C. NOT NULL
8. B. SELECT category, MAX(price) FROM "Menu Items" GROUP BY category;
9. A. It selects the names of customers who have placed orders on or after January 1, 2023.
10. B. SELECT AVG(rating), menu_item_id FROM Reviews GROUP BY menu_item_id;
11. B. SELECT c.first_name, c.last_name, SUM(o.total_amount) FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id GROUP BY c.customer_id;
12. A. It selects the names of customers who have placed orders with a total amount greater than 50.
13. A. SELECT c.first_name, c.last_name FROM Customers c JOIN Orders o ON c.customer_id = o.customer_id GROUP BY c.customer_id HAVING COUNT(o.order_id) >= 2;
14. A. It selects the names of the most expensive menu items in the "Appetizers" category.
15. B. SELECT c.first_name, c.last_name FROM Customers c WHERE c.customer_id IN (SELECT customer_id FROM Orders) AND c.customer_id IN (SELECT customer_id FROM Reservations) AND c.customer_id IN (SELECT customer_id FROM Reviews);
16. A. SELECT COUNT(*) FROM Orders o JOIN "Tables" t ON o.table_number = t.table_number WHERE t.seating_capacity >= 4;
17. D. SELECT name FROM "Menu Items" LEFT JOIN Reviews ON "Menu Items".item_id = Reviews.item_id WHERE Reviews.item_id IS NULL;
18. B. SELECT c.first_name, c.last_name FROM Customers c WHERE c.customer_id IN (SELECT customer_id FROM Orders WHERE order_date >= (SELECT MAX(reservation_date) FROM Reservations WHERE customer_id = c.customer_id));
19. D. SELECT e.first_name, e.last_name FROM Employees e WHERE e.employee_id IN (SELECT p.employee_id FROM Payments p WHERE p.order_id IN (SELECT order_id FROM Orders WHERE total_amount > 1000));

20. C. SELECT COUNT(*) FROM Customers c WHERE c.customer_id IN (SELECT customer_id FROM Orders) INTERSECT SELECT customer_id FROM Reservations;