SQL TEST - Regex Software

Total points 19/25

25 Question - 25 Marks

The respondent's email (**yashsankhla48@gmail.com**) was recorded on submission of this form.

Full Name *

Yash Sankhla

Email *

yashsankhla48@gmail.com

Phone Number *

8209860487

1/1

1. How do you select all columns from a table named "Customers"?

- (●) SELECT * FROM Customers ✓
- ( ) SELECT [all] FROM Customers
- ( ) SELECT Customers
- ( ) EXTRACT * FROM Customers

✓ 1/1

2. Which SQL statement is used to delete data from a database?

- ○ REMOVE
- ◉ DELETE ✓
- ○ DROP
- ○ ERASE

✓ 1/1

3. How can you insert "ID = 5", "FirstName = John", and "LastName = Doe" into the "Users" table?

- ○ INSERT INTO Users VALUES ('John', 'Doe')
- ○ INSERT INTO Users VALUES ('John', 5 'Doe')
- ○ ALTER TABLE ADD INTO Users (ID, FirstName, LastName) VALUES (5, 'John', 'Doe')
- ◉ INSERT INTO Users (ID, FirstName, LastName) VALUES (5, 'John', 'Doe') ✓

✓ 1/1

4. Which statement is used to add a column in an existing table?

- ○ ADD COLUMN
- ◉ ALTER TABLE ADD COLUMN ✓
- ○ MODIFY TABLE
- ○ UPDATE TABLE ADD COLUMN

0/1

5. How can you change the value of the "FirstName" column to "Jane" for all records in the "Users" table where the "LastName" is "Doe"?

- ○ MODIFY Users FirstName='Jane' WHERE LastName='Doe'
- ◉ ALTER Users SET FirstName='Jane' WHERE LastName='Doe' ✗
- ○ UPDATE Users SET FirstName='Jane' WHERE LastName='Doe'
- ○ UPDATE Users SET LastName='Doe' where FirstName='Jane'

Correct answer

- ◉ UPDATE Users SET FirstName='Jane' WHERE LastName='Doe'

✓ 1/1

6. Imagine you work with a database that tracks sales transactions for an online retailer. The database contains several tables, but you are primarily concerned with two of them: Customers and Orders. The Customers table includes columns for CustomerID, CustomerName, ContactName, and Country. The Orders table contains OrderID, CustomerID (which links to the Customers table), OrderDate, and TotalAmount.

You are tasked with generating a report that shows the total amount spent by each customer on their orders. You decide to write an SQL query to fetch this information.

Question
Which of the following SQL queries correctly produces a report showing the total amount spent by each customer, including the customer's name and the total amount spent, sorted by the total amount spent in descending order?

- "SELECT CustomerName, COUNT(TotalAmount) AS TotalSpent FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.OrderID GROUP BY CustomerName ORDER BY TotalSpent DESC;
- SELECT CustomerName, TotalAmount, FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID ORDER BY TotalAmount DESC;
- SELECT CustomerName, SUM(TotalAmount) AS TotalSpent FROM Customers JOIN Orders ON Customers.CustomerID =Orders.CustomerID GROUP BY CustomerName ORDER BY TotalAmount DESC;
- (✓) SELECT CustomerName, SUM(TotalAmount) AS TotalSpent FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID GROUP BY CustomerName ORDER BY TotalSpent DESC;

1/1

7. You are a database analyst working with a database that manages information for a bookstore. The database includes several tables, but you are primarily concerned with two of them: Books and Authors. The Books table contains columns for BookID, Title, Genre, AuthorID, and Price. The Authors table contains AuthorID, AuthorName, Country, and YearOfBirth.

You are tasked with finding all books in the 'Science Fiction' genre that are priced more than 20 units (assuming currency is not specified) and were written by authors from the United States. You need to return the titles of these books along with the names of their authors.

Tables
Books
BookID (int)
Title (varchar)
Genre (varchar)
AuthorID (int)
Price (decimal)
Authors
AuthorID (int)
AuthorName (varchar)
Country (varchar)
YearOfBirth (int)
Question
Which of the following SQL queries correctly retrieves the required information based on the scenario described?

- SELECT Title, AuthorName FROM Books JOIN Authors ON Books.AuthorID = Authors.AuthorID WHERE Genre = 'Science Fiction' AND Price > 20 AND Country = 'United States';
- SELECT Books.Title, Authors.AuthorName FROM Books INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID WHERE Books.Genre = 'Science Fiction' AND Books.Price > 20 AND Authors.Country = 'United States';
- SELECT Title, AuthorName FROM Books, Authors WHERE Books.AuthorID = Authors.AuthorID AND Genre = 'Science Fiction' AND Price > 20 AND Authors.Country = 'USA';

- SELECT Title, AuthorName FROM Books LEFT JOIN Authors ON Books.AuthorID = Authors.AuthorID WHERE Genre = 'Science Fiction' AND Price > 20;

✓ 1/1

8. You have a table named Employees with columns EmployeeID, FirstName, LastName, and DepartmentID. You need to write a query to find all employees who do not belong to any department yet. Which of the following queries will accomplish this?

- SELECT * FROM Employees WHERE DepartmentID = NULL;
- (●) SELECT * FROM Employees WHERE DepartmentID IS NULL ✓
- SELECT * FROM Employees WHERE DepartmentID != NULL;
- SELECT * FROM Employees WHERE DepartmentID IS NOT NULL;

✗ 0/1

9. You are tasked with creating a new table named Orders that includes an OrderID, OrderDate, and CustomerID. Which of the following SQL statements correctly creates this table with OrderID as the primary key?

- CREATE TABLE Orders (OrderID int NOT NULL, OrderDate date, CustomerID int, PRIMARY KEY (OrderID));
- (●) CREATE TABLE Orders (OrderID int PRIMARY KEY, OrderDate date, CustomerID int); ✗
- Both A and B are correct.
- Both are incorrect

Correct answer

- (●) Both A and B are correct.

✓ 1/1

10. A table named Products has a column Prices. You need to increase the price of all products by 10%. Which SQL statement achieves this?

- ○ UPDATE Products SET Price = Price * 1.100;
- ○ ALTER Products SET Price = Price + Price * 0.10;
- ○ MODIFY Products INCREASE Price BY 10%;
- ◉ UPDATE Products SET Prices = Prices * 1.10; ✓

✓ 1/1

11. You want to retrieve a list of employees' full names by concatenating their first and last names, from a table named Employees. Which of the following queries would correctly achieve this in a column named FullName?

- ○ SELECT FirstName + ' ' + LastName AS FullName FROM Employees;
- ○ SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM Employees;
- ◉ Both A and B, depending on the SQL database being used. ✓
- ○ SELECT FullName(FirstName, LastName) FROM Employees;

✓ 1/1

12. A database has a table named Sales with columns SaleDate and Amount. You need to find the total sales (Amount) per year. Which query should you use?

- (●) SELECT YEAR(SaleDate), SUM(Amount) FROM Sales GROUP BY YEAR(SaleDate); ✓
- ( ) SELECT SaleDate, SUM(Amount) FROM Sales GROUP BY SaleDate;
- ( ) SELECT SUM(Amount) FROM Sales WHERE YEAR(SaleDate);
- ( ) SELECT SaleDate, TOTAL(Amount) FROM Sales GROUP BY SaleDate;

✓ 1/1

13. You have a table named CustomerOrders with columns CustomerID, OrderID, and OrderDate. You want to find the most recent OrderDate for each CustomerID. Which SQL statement will provide the correct result?

- ( ) SELECT CustomerID, OrderDate FROM CustomerOrders GROUP BY CustomerID, OrderDate;
- ( ) SELECT MAX(OrderDate), CustomerID FROM CustomerOrders ORDER BY CustomerID;
- ( ) SELECT CustomerID, OrderDate FROM CustomerOrders WHERE OrderDate = MAX(OrderDate);
- (●) SELECT CustomerID, MAX(OrderDate) FROM CustomerOrders GROUP BY CustomerID; ✓

✓ 1/1

14. A table EmployeeDetails contains columns EmployeeID, Name, and HireDate. You need to list all employees who were hired in the last 30 days. Which SQL query is correct?

- (●) SELECT * FROM EmployeeDetails WHERE HireDate > NOW() - INTERVAL 30 DAY; ✓
- ( ) SELECT * FROM EmployeeDetails WHERE HireDate > DATEADD(day, -30, CURRENT_DATE);
- ( ) SELECT * FROM EmployeeDetails WHERE HireDate BETWEEN CURRENT_DATE AND DATEADD(day, -30, CURRENT_DATE);
- ( ) SELECT * FROM EmployeeDetails WHERE HireDate > CURRENT_DATE - 30;

✓ 1/1

15. Assuming a table Inventory with columns ProductID, ProductName, and Quantity, you wish to find all products with a quantity less than 5 for an urgent restock. Which query should you use?

- ( ) SELECT ProductID FROM Inventory WHERE Quantity <= 5;
- ( ) SELECT ProductName FROM Inventory HAVING Quantity < 5;
- (●) SELECT * FROM Inventory WHERE Quantity < 5; ✓
- ( ) SELECT product_id,product_name FROM Inventory WHERE Quantity < 5;

✓ 1/1

16. You are working with a table named CustomerFeedback that includes columns FeedbackID, CustomerID, and Rating (1 to 5). You need to calculate the average rating for each CustomerID. Which SQL query will achieve this?

- ○ SELECT CustomerID, SUM(Rating)/COUNT(CustomerID) FROM CustomerFeedback GROUP BY CustomerID;
- ○ SELECT CustomerID, Rating FROM CustomerFeedback AVERAGE BY CustomerID;
- ○ SELECT AVG(Rating), CustomerID FROM CustomerFeedback;
- ◉ SELECT CustomerID, AVG(Rating) FROM CustomerFeedback GROUP BY CustomerID; ✓

✗ 0/1

17. In a database containing a table named "Students" with columns "StudentID", "FirstName", "LastName", and "GPA", you need to retrieve the names of all students whose GPA is above 3.5. Which SQL query would you use?

- ◉ SELECT * FROM Students WHERE GPA > 3.5 ✗
- ○ SELECT FirstName, LastName FROM Students WHERE GPA > 3.5;
- ○ SELECT Names FROM Students WHERE GPA > 3.5;
- ○ SELECT FirstName, LastName FROM Students HAVING GPA > 3.5;

Correct answer

- ◉ SELECT FirstName, LastName FROM Students WHERE GPA > 3.5;

✓ 1/1

18. In a database with a table named "Books" containing columns "BookID", "Title", "Author", and "PublicationYear", you need to find the number of books published each year. Which SQL query would you use?

- (●) SELECT PublicationYear, COUNT(BookID) FROM Books GROUP BY PublicationYear; ✓
- ( ) SELECT BookID, COUNT(PublicationYear) FROM Books GROUP BY PublicationYear;
- ( ) SELECT COUNT() FROM Books GROUP BY PublicationYear;
- ( ) SELECT PublicationYear, COUNT() FROM Books;

✓ 1/1

19. Imagine a scenario where you have a table named "Orders" with columns "OrderID", "CustomerID", "OrderDate", and "TotalAmount". You need to retrieve the count of orders placed by each customer. Which SQL query would you use?

- ( ) SELECT COUNT(OrderID) FROM Orders;
- (●) SELECT CustomerID, COUNT(OrderID) FROM Orders GROUP BY CustomerID; ✓
- ( ) SELECT COUNT(OrderID),CustomerID FROM Orders GROUP BY CustomerID;
- ( ) SELECT CustomerID, COUNT() FROM Orders;

✓ 1/1

20. Consider a scenario where you have a table named "Products" with columns "ProductID", "ProductName", "CategoryID", and "Price". You need to delete all products belonging to a specific category. Which SQL statement would you use?

- ○ DELETE * FROM Products WHERE CategoryID = 123;
- ◉ DELETE FROM Products WHERE CategoryID = 123; ✓
- ○ REMOVE FROM Products WHERE CategoryID = 123;
- ○ DELETE * FROM Products WHERE CategoryID >= 123;

✗ 0/1

21. In a database with a table named "Students" containing columns "StudentID", "FirstName", "LastName", and "Age", you need to retrieve the youngest student's information. Which SQL query would you use?

- ○ SELECT * FROM Students ORDER BY Age LIMIT 1;
- ○ SELECT * FROM Students WHERE Age = (SELECT MIN(Age) FROM Students);
- ○ SELECT TOP 1 * FROM Students ORDER BY Age ASC;
- ◉ SELECT * FROM Students WHERE Age = MIN(Age); ✗

Correct answer

- ◉ SELECT * FROM Students WHERE Age = (SELECT MIN(Age) FROM Students);

✓ 1/1

22. Scenario: You need to find the total number of orders made by each customer and list them in descending order of the total number of orders. Which SQL query would achieve this?

- (●) SELECT customer_id, COUNT(order_id) FROM Orders GROUP BY customer_id ORDER BY COUNT(order_id) DESC; ✓
- ( ) SELECT customer_id, COUNT(order_id) FROM Orders ORDER BY COUNT(order_id) DESC;
- ( ) SELECT customer_id, SUM(order_id) FROM Orders GROUP BY customer_id ORDER BY SUM(order_id) DESC;
- ( ) SELECT customer_id, COUNT() FROM Orders GROUP BY customer_id ORDER BY COUNT() DESC;

✗ 0/1

23. Scenario: You want to find the total sales amount for each month of the year 2023. Which SQL query would you use?

- ( ) SELECT MONTH(order_date), SUM(sales_amount) FROM Orders GROUP BY MONTH(order_date) WHERE YEAR(order_date) = 2023;
- (●) SELECT MONTH(order_date), SUM(sales_amount) FROM Orders WHERE YEAR(order_date) = 2023 GROUP BY MONTH(order_date); ✗
- ( ) SELECT MONTH(order_date), SUM(sales_amount) FROM Orders GROUP BY YEAR(order_date), MONTH(order_date) HAVING YEAR(order_date) = 2023;
- ( ) SELECT MONTH(order_date), SUM(sales_amount) FROM Orders WHERE order_date BETWEEN '2023-01-01' AND '2023-12-31' GROUP BY MONTH(order_date);

Correct answer

- (●) SELECT MONTH(order_date), SUM(sales_amount) FROM Orders WHERE order_date BETWEEN '2023-01-01' AND '2023-12-31' GROUP BY MONTH(order_date);

✓ 1/1

24. Consider a database with two tables: Employees and Departments. Each employee is assigned to a department, and each department has a unique department ID. Which of the following SQL queries will retrieve the names of all employees who belong to the "Sales" department?

- (●) SELECT Name FROM Employees WHERE DepartmentID = (SELECT DepartmentID FROM Departments WHERE DepartmentName = 'Sales'); ✓
- ( ) SELECT Name FROM Employees WHERE DepartmentName ='Sales';
- ( ) SELECT Name FROM Employees WHERE DepartmentID IN (SELECT DepartmentID FROM Departments WHERE Department Name = 'Sales');
- ( ) SELECT Name FROM Employees WHERE DepartmentID EXISTS (SELECT DepartmentID FROM Departments WHERE DepartmentName = 'Sales');

✗ 0/1

25. Suppose you need to retrieve the names of all employees who have a salary higher than the average salary in their respective departments. Which of the following SQL queries will achieve this?
Both Tables are different for Employee and Department

- (●) SELECT Name FROM Employees WHERE Salary > AVG(Salary) GROUP BY DepartmentID; ✗
- ( ) SELECT Name FROM Employees WHERE Salary >(SELECT AVG(Salary)FROM Employees GROUP BY DepartmentID);
- ( ) SELECT Name FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees) GROUP BY DepartmentID;
- ( ) SELECT Name FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees WHERE Employees.DepartmentID =Departments.DepartmentID);

Correct answer

- (●) SELECT Name FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees WHERE Employees.DepartmentID =Departments.DepartmentID);