SQL Assessment

1. When do you use select, update, insert, delete, truncate?

SELECT retrieves records.

UPDATE modifies records.

INSERT adds records to a table.

DELETE removes records.

TRUNCATE acts on an entire table. It removes all record data while keeping the structure of the table, unlike DROP which deletes the entire table.

1. How do you define and use a subquery?

Subqueries are complete queries that appear in parentheses within a larger query. They execute before the main query, so the results of the subquery can be used as a component in the larger query. This can be useful for, for example, comparisons, or doing a calculation on a subset of records.

1. When do you use where vs. having statements?

WHERE and HAVING both provide conditions for data. The difference is whether the data to be retrieved is individual records or grouped records. WHERE statements provide a condition for individual records. HAVING applies a condition to a GROUP BY clause. WHERE executes before the GROUP BY clause, while HAVING executes after GROUP BY.

1. When is a group by required?

GROUP BY is required any time multiple records need to be combined, often to use with aggregate functions like SUM() or AVG(). For example, if you want to find the average salary for each department, you can use:

SELECT Department, AVG(Salary) AS Avg\_Salary

FROM Employees

GROUP BY Department;

1. Describe the differences between Inner, Right, Left, Full, and Cross Joins

Inner joins only return records that have matches in both Table A and Table B. Full joins return all records from both Table A and Table B, regardless of whether they have a match in the other table. Right or left inner joins return all the mutual matches between Tables A and B as well as all the records from either the right or left table. Cross joins combine each row from Table A with each row from Table B.

1. Based on the attached table structure, write a query to return the last 5 (by order date) order ids from the orders table.

SELECT orderid, orderdate

FROM Orders

ORDER BY orderdate DESC

LIMIT 5;

1. Based on the attached table structure, write a query to return, by month, the unique number of events that had orders between 1/1/2020 and 12/31/2020 inclusive.

SELECT MONTH(Event.eventdate) AS Month, COUNT(\*) AS Num\_Events

FROM Event

INNER JOIN Orders

ON Event.eventid = Orders.eventid

WHERE Orders.orderdate BETWEEN ‘01/01/2020’ AND ‘12/31/2020’

GROUP BY MONTH(Event.eventdate);

1. Based on the attached table structure, write a query to return the total revenue, count of unique orders by eventname for events that occurred between 3/15/2020 and 4/15/2020 inclusive.

SELECT COUNT(\*), SUM(Orders.ordertotal) AS Revenue

FROM Event

INNER JOIN Orders

ON Event.eventid = Orders.eventid

WHERE Event.evendate BETWEEN ‘03/15/2020’ AND ‘04/15/2020’

GROUP BY Event.eventname;

1. Based on the attached table structure, write a query to return a list of customerids that have placed multiple orders between 3/15/2020 and 4/15/2020 inclusive.

SELECT Customer.customerid, COUNT(\*)

FROM Customer

INNER JOIN Orders

ON Customer.customerid = Orders.customerid

WHERE Orders.orderdate BETWEEN ‘03/15/2020’ AND ‘04/15/2020’

GROUP BY Customer.customerid

HAVING COUNT(\*) > 1;

1. Based on the attached table structure, return the email address of the customer along with the event name and order date of the last event the customer purchased

WITH Last\_Order AS

(SELECT Orders.customerid AS Customer, MAX(Orders.orderdate) AS Latest\_Order\_Date,

Orders.emailaddress AS Email

FROM Orders

GROUP BY Orders.customerid)

SELECT Email, Latest\_Order\_Date, Event.eventname

FROM Last\_Order

INNER JOIN Event

ON Last\_Order.eventid = Event.eventid;

1. Based on the attached table structure write a query to return the number of orders placed during the on sale date vs. non-on sale dates for each event in January 2021

SELECT Event.eventname,

COUNT(CASE WHEN Orders.orderdate = Event.onsaledate THEN \* END) AS Num\_On\_Sale\_Date,

COUNT(CASE WHEN Orders.orderdate != Event.onsaledate THEN \* END) AS Non\_Sale\_Date

FROM Event

INNER JOIN Orders

ON Event.eventid = Orders.eventid;