**Review Questions:** 1. Compare single-row functions, multiple-row functions, and analytic functions in terms of required data source and their output. Single-Row Functions Source Data: Single Row Output: One row of result Aggregate Functions Source Data: Multiple rows Analytic Functions Source Data: Multiple Rows Output: Multiple rows of results2. SQL analytic functions compute an aggregate value based on a group of rows. The group of rows is called a partition.The analytic functions allow users to divide query result sets into groups of rows called partition.3. Analytic functions can appear in the SELECT clause of a query.4. A cumulative sum is a sequence of partial sums of a given sequence. For example, the cumulative sums of the sequence (a, b, c, …) are a, a+b, a + b +5. In a windowing clause, the keyword ROWS is used to specify the window in physical units.6. In a windowing clause, the keyword RANGE is used to specify the window as a logical interval.7. A mean is an average value for a variable. In other words, it is the sum of the data points divided by the number of data points. It is that value that is most commonly referred to as the average.8. A median is the value in the middle when the data are arranged in ascending order.9. A mode is the value that occurs most frequently in a data set.10. Given a series of data on students’ GPA: 2.8, 3.0, 3.0, 3.0, 3.2, 3.7, 3.8, 4.0Mean = 3.3125Median = 3.0Mode = 3.0 **TRUE/FALSE** Determine which of the statements is true? Which is false? Why are they false?The window determines the range of rows used to perform the calculations or other aggregate functions for the current row. TrueWindow sizes can be based on either a physical number of rows or a logical interval such as time. TrueIn case of defining a window for a moving average function, a window will have both its starting and end points slide so that the window maintains a constant physical or logical range. TrueA query result set may be partitioned into just one partition holding all the rows, a few large partitions, or many small partitions holding just a few rows each. TrueThe window size can possibly be variable in some cases. TrueA window can be set as large as all the rows in a partition or just a sliding window of one row within a partition. TrueIf the PARTITION BY clause is omitted in an analytic function, then the function treats all rows of the query result set as a single group. TrueThe window size should always be a constant (i.e., fixed) in any case. False because it is the opposite of the above statementWhen a window is near a border, the function will not return results but warn you that the results are not what you want. FalseIn case of defining a window for a cumulative sum function, the starting point of the window should always be at the current row of its partition, and its end point would slide from the starting point all the way to the last row of the partition. False (the first of its partition)UNBOUNDED PRECEDING can only be specified as a window end point. False (starting point)UNBOUNDED FOLLOWING can only be specified as a window starting point. It cannot be used as an end point specification. False (starting point)The ORDER BY clause in an analytic function is used to specify how data is ordered within a partition. We can order the values in a partition on a single column only. FalseWe can use the PARTITION BY clause to partition the query result set into groups based on a single column only. False **COMPLETING SQL CODE** 1. Complete the following query that will produce OUTPUT A: SELECT firstname || ' ' || lastname employee, monthly\_salary,SUM (monthly\_salary) OVER (ORDER BY lastname, firstname ROWS BETWEEN 1 PRECEDING AND UNBOUNDED FOLLOWING) sum FROM emp; 2. Complete the following query that will produce OUTPUT B: SELECT firstname || ' ' || lastname employee, monthly\_salary, SUM (monthly\_salary) OVER (ORDER BY lastname, firstname ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) sum FROM emp; 3. Complete the following query that will produce OUTPUT C: SELECT firstname || ' ' || lastname employee, monthly\_salary, SUM (monthly\_salary) OVER (ORDER BY lastname, firstname ROWS UNBOUNDED PRECEDING) sum FROM emp; 4. Complete the following analytic query that will produce the output as shown above: SELECT lastname, firstname, deptno, hiredate, SUM OF monthly\_salary, OVER (PARTITION BY deptno ORDER BY hiredate ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) dept\_total FROM emp; 5. Complete the following analytic query that will provide a summary of each employee’s salary together with the salary values of all employees whose HIRE\_DATE value falls within one week (7 days) preceding the current row.” SELECT lastname, firstname, deptno, hiredate, monthly\_salary, SUM (monthly\_salary) OVER (ORDER BY hiredate RANGE BETWEEN 7 PRECEDING AND CURRENT ROW) "one\_week\_total" FROM emp SELECT lastname, firstname, deptno, hiredate, monthly\_salary, SUM (monthly\_salary) OVER (ORDER BY hiredate RANGE 7 PRECEDING) "one\_week\_total" FROM emp 6. Complete the following analytic query to compute a cumulative average of annual pay (excluding commission) for all the employees in the company (as shown above). SELECT lastname, firstname, AVG(monthly\_salary^12)OVER(ORDER BY lastname) cumulative\_avg FROM emp; 7. Complete the following analytic query to compute a cumulative average of annual pay (excluding commission) for the employees in each department (as shown above).

SELECT lastname, firstname, AVG (monthly\_salary\*12) OVER \_\_\_\_\_\_ running\_avg

FROM emp ORDER BY lastname;

**QUESTIONS ON WINDOWING CLAUSES**

1, How the window is defined with the following specification (in terms of the window size, starting point, and end point)?

* ROWS/RANGE 2 PRECEDING
* ROWS/RANGE BETWEEN 2 PRECEDING AND CURRENT ROW
* ROWS/RANGE BETWEEN CURRENT ROW AND 2 FOLLOWING
* ROWS/RANGE BETWEEN 1 PRECEDING AND 2 FOLLOWING
* ROWS/RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

2. What does the following query do? Is it executable? If not, why? SELECT firstname || ' ' || lastname employee, monthly\_salary, SUM (monthly\_salary) OVER (ORDER BY lastname, firstname ROWS BETWEEN CURRENT ROW AND UNBOUNDED PRECEDING) sum FROM emp; 3. Which of the following windowing clauses will always prompt an error message? WHY? a. ROWS BETWEEN UNBOUNDED FOLLOWING AND CURRENT ROW f. ROWS BETWEEN CURRENT ROW AND 2 PRECEDING g. ROWS BETWEEN 2 FOLLOWING AND CURRENT ROW 4. Which of the following analytic queries is syntactically valid and truly addresses the question to show a cumulative sum of monthly salary for all the employees in the company? c. SELECT lastname, firstname, monthly\_salary,

SUM (monthly\_salary) OVER (ORDER BY lastname, firstname) running\_sum FROM emp ORDER BY lastname; 5. The figure as shown above is the outcome produced from running the following analytic query. SELECT lastname, firstname, commission, SUM (commission) OVER (ORDER BY lastname, firstname) running\_total FROM emp WHERE jobtitle = 'SALES REP.' ORDER BY lastname; Then x1 = 300 x2 = 1700 x3 = 2700 x4 = 3200 6. The figure as shown above is the outcome produced from running the following analytic query. Then x1 = 2200 How about x2 8700 x3 21175 x4 26715 x5 29715 SELECT deptno, lastname, firstname, monthly\_salary, SUM (monthly\_salary) OVER (PARTITION BY deptno ORDER BY lastname, firstname) dept\_total FROM emp

ORDER BY deptno, lastname, firstname; 7. The figure as shown above is the outcome produced from running the following analytic query. Then Then x1 = 3500, x2 40000 x3 5200 x4 8540 SELECT deptno, lastname, firstname, monthly\_salary, SUM (monthly\_salary) OVER (PARTITION BY deptno ORDER BY monthly\_salary ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) dept\_total2 FROM emp ORDER BY deptno;