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 results 2. 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, 4.0 Mean = 3.3125 Median = 3.0 Mode = 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. True Window sizes can be based on either a physical number of rows or a logical interval such as time. True In 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. True A 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. True The window size can possibly be variable in some cases. True A window can be set as large as all the rows in a partition or just a sliding window of one row within a partition. True If 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. True The window size should always be a constant (i.e., fixed) in any case. False because it is the opposite of the above statement When a window is near a border, the function will not return results but warn you that the results are not what you want. False In 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. False We 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; QUESTIONS ON WINDOWING CLAUSES ROWS – specifies the window in physical units (rows) RANGE – specifies the window as a logical offset (interval) n PRECEDING – get n rows before the current one.n FOLLOWING – get n rows after the current one.UNBOUNDED – when used with PRECEDING or FOLLOWING, it returns all before or after.UNBOUNDED PRECEDING – get all rows before the current row UNBOUNDED FOLLOWING – get all rows after the current row 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; Query processing using analytic functions takes place in 4 stages:SELECT and FROM clauses are first performed. All joins, WHERE, GROUP BY and HAVING clauses are then performed. The result set is made available to the analytic functions (applied to each row in each partition), and all their calculations take place.If the query has an ORDER BY clause at its end, the ORDER BY is processed to allow for precise output ordering.Moving Average/Mean (MA/MM) Time-series data can be inherently noisy and a good way to smooth out the data is to calculate a moving average. In statistics, a moving average (moving mean) (or called rolling average/mean or running average/mean) is a calculation to analyze data points by creating series of averages of different subsets of the full data set.Given a series of numbers and a fixed subset size, the first element of the moving average is obtained by taking the average of the initial fixed subset of the number series. Then the subset is modified by "shifting forward"; that is, excluding the first number of the series and including the next number following the original subset in the series. This creates a new subset of numbers, which is averaged. This process is repeated over the entire data series. Windowing Functions A window function performs a calculation across a set of rows that are somehow related to the current row. This is comparable to the type of calculation that can be done with an aggregate function. But unlike regular aggregate functions, use of a window function does not cause rows to become grouped into a single output row — the rows retain their separate identities. Behind the scenes, the window function is able to access more than just the current row of the query result. SUM Windowing For each row in a partition, you can define a sliding window of data. This window determines the range of rows (i.e., window sizes) used to perform the calculations for the current row. Window sizes can be based on either a physical number of rows (e.g., preceding three rows) or a logical interval such as a range of time periods (e.g., those records involved within the past or next 6 months). Window Sizes A window can be set as large as all the rows in a partition or as a sliding window of one row within a partition. When a window is near a border, the function returns results for only the available rows, rather than warning you that the results are not what you want. Windowing

Image result for sliding window The window has a starting row and an ending row. In some cases, a window defined for a cumulative sum function would have its starting row fixed at the first row of its partition all the time, and its ending row would slide from the starting point all the way to the last row of the partition. In contrast, a window defined for a moving average would have both its starting and end points slide so that they maintain a constant physical or logical range.Depending on its definition, the window may move at one or both ends. ROWS Specifying the window in physical units (rows) ROWS n PRECEDING ROWS n PRECEDING (= ROWS BETWEEN n PRECEDING AND CURRENT ROW) specifies the aggregate functions in the current partition in the OVER clause that includes the current row, plus a specific number of rows before the current row. UNBOUNDED PRECEDING UNBOUNDED PRECEDING to indicate that the sliding window always starts at the first row of the partition.Starting Point: UNBOUNDED PRECEDING can only be specified as a window starting point. It cannot be used as an end point specification.End Point: As an end point, CURRENT ROW is the end point specifying that the window ends at the current row or value. LAST\_VALUE LAST\_VALUE is an analytic function returning the last value in an ordered set of values. If the last value in the set is null, then the function returns NULL unless you specify IGNORE NULLS. If you specify IGNORE NULLS, then LAST\_VALUE returns the fist non-null value in the set, or NULL if all values are null. UNBOUNDED FOLLOWING UNBOUNDED FOLLOWING to indicate that the window always ends at the last row of the partition. UNBOUNDED FOLLOWING can only be specified as a window end point (but not the start point) For example ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING defines a window that starts with the current row and ends with the last row of the partition. As a start point, CURRENT ROW specifies that the window begins at the current row or value (depending on whether you have specified ROWS or RANGE, respectively). Last Row of the Partition as the (Fixed) End Point The window is decreasing in size.

UNBOUNDED FOLLOWING is the end point specification and cannot be used as a start point specification. FIRST\_VALUE

FIRST\_VALUE is an analytic function. It returns the first value in an ordered set of values. If the first value in the set is null, then the function returns NULL unless you specify IGNORE NULLS. If you specify IGNORE NULLS, then FIRST\_VALUE returns the fist non-null value in the set, or NULL if all values are null. TIP – ORDER BY cannot be used in computing MEDIAN STATS\_MODE – Computing Statistical Mode STATS\_MODE takes as its argument a set of values and returns the value that occurs with the greatest frequency.