Module 3 Assignment

The module 3 assignment provides experience writing SELECT statements using the SQL analytic functions. You should adapt the examples given in the notes. Each problem is based on a similar problem in the notes.

Your SELECT statements will reference the tables of the Inventory Data Warehouse, described in another document. The INSERT statements are provided in another document. The Inventory Data Warehouse design and rows are identical from module 5 in course 2. If you added rows through the data integration assignment in module 5 of course 2, you should remove those rows or just recreate and repopulate the tables.

Query 1: Ranking within the entire result

Use the RANK function to rank customers in descending order by the sum of extended cost for shipments (transaction type 5). You should use the entire result as a single partition. The result should include the customer name, sum of the extended cost, and rank.

Query 2: Ranking within a partition

Use the RANK function to rank customers in descending order by the sum of extended cost for shipments (transaction type 5). You should partition the rank values by customer state. The result should include the customer state, customer name, sum of the extended cost, and rank. You should order the result by customer state.

Query 3: Ranking and dense ranking within the entire result

Use both RANK and DENSE_RANK functions to rank customers in descending order by the count of inventory transactions for shipments (transaction type 5). You should use the entire result as a single partition. The result should include the customer name, count of transactions, rank, and dense rank.

Query 4: Cumulative extended costs for the entire result

Calculate the cumulative sum of extended cost ordered by customer zip code, calendar year, and calendar month for shipments (transaction type 5). The result should include the customer zip code, calendar year, calendar month, sum of the extended cost, and cumulative sum of the extended cost. Note that the cumulative extended cost is the sum of the extended cost in the current row plus the cumulative sum of extended costs in all previous rows.

Query 5: Cumulative extended costs for a partition

Calculate the cumulative sum of extended cost ordered by customer zip code, calendar year, and calendar month for shipments (transaction type 5). Restart the cumulative extended cost after each combination of zip code and calendar year. The result should include the customer zip code, calendar year, calendar month, sum of the extended cost, and cumulative sum of the extended cost. Note that the cumulative extended cost is the sum of the extended cost in the current row plus the cumulative sum of extended costs in all previous rows of the store zip code and years. The value of cumulative extended cost resets in each partition (new value for zip code and year).

Query 6: Ratio to report applied to the entire result

Calculate the ratio to report of the sum of extended cost for adjustments (transaction type 1). You should sort on descending order by sum of extended cost. The result should contain the second item id, sum of extended cost, and ratio to report.

Note: Since PostgreSQL does not support the RATIO_TO_REPORT function, learners using PostgreSQL have the choice of submitting an Oracle SQL statement without a snapshot to show execution or a substitute PostgreSQL statement to produce identical results with an alternative PostgreSQL statement. You should see the slides in lesson 5 of module 3 about a query pattern without the RATIO_TO_REPORT analytic function.

Query 7: Ratio to report applied to a partition

Calculate the ratio to report of the sum of extended cost for adjustments (transaction type 1) with partitioning on calendar year. You should sort on ascending order by calendar year and descending order by sum of extended cost. The result should contain the calendar year, second item id, sum of extended cost, and ratio to report.

Note: Since PostgreSQL does not support the RATIO_TO_REPORT function, learners using PostgreSQL have the choice of submitting an Oracle SQL statement without a snapshot to show execution or a substitute PostgreSQL statement to produce identical results with an alternative PostgreSQL statement. You should see the slides in lesson 5 of module 3 about a query pattern without the RATIO_TO_REPORT analytic function.

Query 8: Cumulative distribution functions for carrying cost of all branch plants

Calculate the rank, percent_rank, and cume_dist functions of the carrying cost in the branch_plant_dim table. The result should contain the BPName, CompanyKey, CarryingCost, rank, percent_rank, and cume_dist.

Query 9: Determine worst performing plants

Determine the branch plants with the highest carrying costs (top 15%). The result should contain BPName, CompanyKey, CarryingCost, and cume_dist.

Query 10: Cumulative distribution of extended cost for Colorado inventory

Calculate the cumulative distribution of extended cost for Colorado inventory (condition on customer state). The result should contain the extended cost and cume_dist, ordered by extended cost. You should eliminate duplicate rows in the result.

Grading

Your performance will be assessed by a quiz designed to test your understanding of each problem and by evidence of query executions. Since some quiz questions involve execution results, you should execute your statements using the original inventory data warehouse tables.

- You will receive 50% if your documentation contains a SELECT statement and partial results for each problem. Execution of your SQL statements demonstrates correct syntax.
- The quiz score will provide the other 50% of your grade. The quiz contains questions about the important elements of each problem such as the usage of the correct tables, function, partitioning, and sorting.

Completion

You should upload a file containing your Oracle SQL statements and execution results to the graded item in module 2. For the execution results, you should take a snapshot of the script output window in SQL Developer showing the execution results. You only need to show the first 10 rows or so of the result. You should paste the execution results after the associated SELECT statement.