Chapter 5 Part 2

How to code summary queries

Exercises

1. **(10 Points)** Write a SELECT statement that uses the GROUP BY ROLLUP clause to return the count of orders for each State and City, with summary rows for the non-aggregate columns (ROLLUP automatically does this):

   The **State** column from the Addresses table

   The **City** column from the Addresses table

   A count of the number of orders

   Sort the result set in descending sequence by **State**, and descending sequence by **City**.

2. **(10 Points)** Write a SELECT statement that uses the GROUP BY CUBE clause to return the count of orders for each State and City, with summary rows for the non-aggregate columns (CUBE automatically does this):

   The **State** column from the **Addresses** table

   The **City** column from the **Addresses** table

   A count of the number of orders

   Sort the result set in descending sequence by **State**, and descending sequence by **City**.

3. **(20 Points)** Write a SELECT statement that uses the GROUP BY GROUPING SETS clause to return the count of orders for each customer (using concatenation of First and Last name), as well as the count of orders for State and City, with summary rows for the non-aggregate columns (using ROLLUP):

   A calculated column with alias "Fullname" that contains the first and last name of the customer with a space in between first and last name.

   The **State** column from the **Addresses** table

   The **City** column from the **Addresses** table

   A count of the number of orders

   Sort the result set in descending sequence by **State**, and descending sequence by **City**.

   Make sure the final summary row is in the correct position (this can be adjusted with the order of the items in the GROUPING SET).

4. **(20 Points)** Write a SELECT statement that uses the OVER clause to return the sum and average of all order items final prices (ItemPrice minus DiscountAmount) and a count of all the order items. Partition the OVER clause by OrderDate from the Orders table for the sum, average, and count. Also include the product name, orderdate, and the final price for each item (without aggregation) using the

formula (ItemPrice minus DiscountAmount). The select statement should include the following columns in the order below:

- The **ProductName** column from the **Products** table.
- The **OrderDate** column from the **Orders** table.
- A calculated column with alias "ItemSalesPrice" which is calculated as follows: **ItemPrice** minus the **DiscountAmount** from the **OrderItems** table.
- OVER clause: sum of the **ItemPrice** minus the **DiscountAmount** from the **OrderItems** table partitioned by the **OrderDate** from the **Orders** table. Use alias "TotalSalesOnDate" for the calculated column.
- OVER clause: average of the **ItemPrice** minus the **DiscountAmount** from the **OrderItems** table partitioned by the **OrderDate** from the **Orders** table. Use alias "AvgSalesOnDate" for the calculated column.
- OVER clause: count of the **ItemPrice** from the **OrderItems** table partitioned by the **OrderDate** from the **Orders** table. Use alias "TotalItemsSoldOnDate" for the calculated column.

Sort the result set in ascending sequence by **OrderDate** and **ProductName**.