Programming Applications with Databases

Exercise Set 1

This Exercise Set is intented to help you understand the SQL basics. All exercises should be based on the AdventureWorksLT example database.

1. Prepare a query which returns a list of cities where goods have been already delivered based on SalesOrderHeader.ShipToAddressID. Results should be sorted and with eliminated duplicates.
   [1p]
2. Prepare a query which returns two columns: product model name (ProductModel.Name) and the number of products for the model, however, only items for with the number greater than 1 should be included. Please explain the consequences in case the name would be selected as a grouping value.
   [1p]
3. Prepare a query which returns three columns: city name (table Address), number of customers from the city, number of SalesPersons supporting customers from the city.
   [1p]
4. Product categories constitutes a tree. We can expect that all products are assigned to categories which are leaves in the tree. Prepare a query which returns two columns: category name and product name for products which are assigned to categories which are *not* leaves in the tree. If needed, please prepare appropriate test data.
   [1p]
5. Create a report based on SalesLT.SalesOrderHeader and SalesLT.SalesOrderDetail tables which meets the following requirements:

   - Each row of the report corresponds to a specific order expressed by SalesOrderID, SalesOrderNumber, PurchaseOrderNumber columns.
   - Next, based on SalesLT.SalesOrderDetail for each order calculate the total sum of the order with and without applied discounts. *Hint: Look at the LineTotal column. What is its definition?*
   - Finally, add a column representing the amount of the whole order.

   Now, prepare the report in the following variants: **(a)** with all orders, **(b)** with order with the highest total discount (apply reasonable threshold based on the obtained data).
   [1p]
6. Prepare a query which returns three columns: last name and first name of a customer (Customer) and amount of money that the customer saved thanks to the received discounts (SalesOrderDetail.UnitPriceDiscount).
   [1p]
7. Create a table Test with the IDENTITY column where identifier values should start with 1000 and increment by 10. Show the difference between @@IDENTITY i IDENT_CURRENT.
   [1p]
8. Get familiar with the constraint SalesOrderHeader.CK_SalesOrderHeader_ShipDate and show its create statement. Try to add a (or modify existing) data row which violates the constraint. Show the results. Then turn off the constraint and try again. Finally, turn on the constraint again and list the current violations.
   [1p]
9. Add a column CreditCardNumber to the Customer table and for 3 random rows in the SalesOrderHeader table set any value for CreditCardApprovalCode column. Then for customers (Customer) with orders (SalesOrderHeader) where CreditCardApprovalCode value is set, change the CreditCardNumber value for 'X'. Show all respective queries to related to the exercise.
   [1p]
10. Create tables *M1(K INT, V VARCHAR(20))* and *S1(K INT, MFK INT, V VARCHAR(20))* where *K* is a primary key and *MFK* is a foreign key to the table *M*. Now, create tables *M2* and *S2* where the only difference is that *M2* has a primary key based on two columns: *K1* and *K2* and *S2* has an appropriate foreign key. Add some test data, check whether foreign key contraint is working properly. Finally, add *ON UPDATE* and *ON DELETE* clauses and show the difference where different values *NO ACTION*, *SET NULL* or *CASCADE* are introduced.
    [1p]

Remark: in case someone is not able to complete the exercise by a single query, more queries are allowed with introducing additional intermediate tables.

Paweł Rajba