Lab 9: Modifying Data

Inserting Products (1)

Each AdventureWorks product is stored in the `SalesLT.Product` table, and each product has a unique `ProductID` identifier, which is implemented as an `IDENTITY`column in the `SalesLT.Product` table.
Products are organized into categories, which are defined in the `SalesLT.ProductCategory` table.
The products and product category records are related by a common `ProductCategoryID` identifier, which is an `IDENTITY` column in the `SalesLT.ProductCategory` table.
The new product to be inserted is shown in this table:

Name	ProductNumber	StandardCost	ListPrice	ProductCategoryID	SellStartDate
LED Lights	LT-L123	2.56	12.99	37	*<Today>*

IMPORTANT NOTE before you start this chapter: - In order to run all separate queries in a SQL script, make sure to end each statement with a `;`. - DataCamp will only return the result of the last query in your script. - Your database-altering statements will not persist, so you can keep experimenting over and over.

Instructions

- AdventureWorks has started selling the new product shown in the table above. Insert it into the `SalesLT.Product` table, using default or `NULL` values for unspecified columns.
- Once you've inserted the product, run `SELECT SCOPE_IDENTITY();` to get the last identity value that was inserted.
- Add a query to view the row for the product in the `SalesLT.Product`table.

```
-- Finish the INSERT statement

INSERT INTO SalesLT.Product (Name, ProductNumber, StandardCost, ListPrice, ProductCategoryID,
SellStartDate)

VALUES

('LED Lights', 'LT-L123', 2.56, 12.99, 37, GETDATE());


-- Get last identity value that was inserted

SELECT SCOPE_IDENTITY();
```

```
-- Finish the SELECT statement

SELECT * FROM SalesLT.Product

WHERE ProductID = SCOPE_IDENTITY();
```

Inserting Products (2)

AdventureWorks is adding a product category for 'Bells and Horns' to its catalog. The parent category for the new category is 4 (Accessories).

Instructions

- Write a query to insert this new product category into the `SalesLT.ProductCategory` table. Insert the `ParentProductCategoryID`, followed by the `Name` of the new product.
- If you want, you can use a `SELECT` statement afterwards to see if the `SalesLT.ProductCategory` was properly updated.

```
-- Insert a product category

INSERT INTO SalesLT.ProductCategory (ParentProductCategoryID, Name)

VALUES

(4, 'Bells and Horns');
```

Inserting Products (3)

The code from the previous exercise to insert the product category is already included. This new category includes the following two new products.

Name	ProductNumber	StandardCost	ListPrice	ProductCategoryID	SellStartDate
Bicycle Bell	BB-RING	2.47	4.99	*<The new ID for Bells and Horns>*	*<Today>*
Bicycle Horn	BB-PARP	1.29	3.75	*<The new ID for Bells and Horns>*	*<Today>*

Can you add these products?

Instructions

- Insert the two new products with the appropriate `ProductCategoryID`value, based on the product details above.
- Finish the query to join the `SalesLT.Product` and `SalesLT.ProductCategory` tables. That way, you can verify that the data has been inserted. Make sure to use the aliases provided, and default column names elsewhere.

```
-- Insert product category

INSERT INTO SalesLT.ProductCategory (ParentProductCategoryID, Name)

VALUES

(4, 'Bells and Horns');


-- Insert 2 products

INSERT INTO SalesLT.Product (Name, ProductNumber, StandardCost, ListPrice, ProductCategoryID,
SellStartDate)

VALUES

('Bicycle Bell', 'BB-RING', 2.47, 4.99, IDENT_CURRENT('SalesLT.ProductCategory'), GETDATE()),

('Bicycle Horn', 'BB-PARP', 1.29, 3.75, IDENT_CURRENT('SalesLT.ProductCategory'), GETDATE());


-- Check if products are properly inserted

SELECT c.Name As Category, p.Name AS Product

FROM SalesLT.Product AS p

JOIN SalesLT.ProductCategory as c ON p.ProductCategoryID = c.ProductCategoryID

WHERE p.ProductCategoryID = IDENT_CURRENT('SalesLT.ProductCategory');
```

Updating Products

You have inserted data for a product, but the pricing details are not correct. You must now update the records you have previously inserted to reflect the correct pricing.

Instructions

- The sales manager at AdventureWorks has mandated a 10% price increase for all products in the Bells and Horns category. Update the rows in the `SalesLT.Product` table for these products to increase their price by 10%.

- If you want, you can use a `SELECT` statement afterwards to see if the records were properly updated, but we won't check that.

```
-- Update the SalesLT.Product table

UPDATE SalesLT.Product

SET ListPrice = ListPrice * 1.1

WHERE ProductCategoryID =

 (SELECT ProductCategoryID FROM SalesLT.ProductCategory WHERE Name = 'Bells and Horns');
```

Updating Products (2)

The new LED lights you inserted in the previous challenge are to replace all previous light products.

Instructions

- Update the `SalesLT.Product` table to set the `DiscontinuedDate` to today's date for all products in the Lights category (`ProductCategoryID` 37) other than the LED Lights product you inserted previously.
- If you want, you can use a `SELECT` statement afterwards to see if the records were properly updated, but we won't check that.

```
UPDATE SalesLT.Product

SET DiscontinuedDate = GETDATE()

WHERE ProductCategoryID = 37 AND ProductNumber <> 'LT-L123';
```

Deleting Products

The Bells and Horns category has not been successful and it must be deleted from the database.

Instructions

- Delete the records for the Bells and Horns category and its products. You must ensure that you delete the records from the tables in the correct order to avoid a foreign-key constraint violation.
- If you want, you can use a `SELECT` statement afterwards to see if the rows were properly deleted, but we won't check that.

```
DELETE FROM SalesLT.Product

WHERE ProductCategoryID =
```

```
        (SELECT ProductCategoryID FROM SalesLT.ProductCategory WHERE Name = 'Bells and Horns');


-- Delete records from the SalesLT.ProductCategory table
DELETE FROM SalesLT.ProductCategory
WHERE ProductCategoryID =
        (SELECT ProductCategoryID FROM SalesLT.ProductCategory WHERE Name = 'Bells and Horns');
```