Highlights:

* To create tables given their requirement specification
* To use IDENTITY in a column

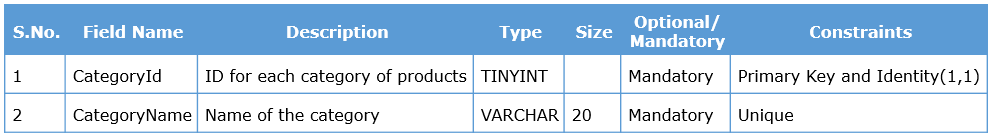
Demosteps:

**Step 1:**

You can create the table Categories for the following specification:

**Name**            :  Categories

**Description**:  It is used for storing the category details of various products



**Step 2:**

To create table Categories, use the **CREATE TABLE**statement.

1. CREATE TABLE Categories

**Step 3:**

Add columns for Category ID and Category Name. Specify the data type of the columns and the constraints applicable on the columns. Use IDENTITY to generate CategoryId for every category by specifying it as an Identity column.

1. CREATE TABLE Categories
2. (
3. CategoryId TINYINT CONSTRAINT pk\_CategoryId PRIMARY KEY IDENTITY,
4. CategoryName VARCHAR(20) CONSTRAINT uq\_CategoryName UNIQUE NOT NULL
5. )
6. GO

**NOTE :** The Seed value and Incrementing value for the Identity column are set to default(1) in this case.

An IDENTITY column must be of one of the following data types

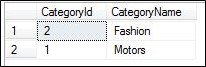
* TINYINT, SMALLINT, INT, BIGINT
* DECIMAL, NUMERIC with a scale of 0

**Step 4:**

Insert two rows into the table Categories and execute the SELECT query to see the results.

1. INSERT INTO Categories VALUES ('Motors')
2. INSERT INTO Categories VALUES ('Fashion')
3. SELECT \* FROM Categories

**Output:**

**

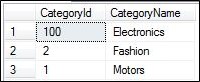
If there is a need to insert the value of CategoryId explicitly, the Identity column value can be explicitly inserted using the syntax shown below:

SET IDENTITY\_INSERT <table\_name> ON

Execute the following and observe the output:

SET IDENTITY\_INSERT Categories ON  
INSERT INTO Categories (CategoryId, CategoryName) VALUES (100, 'Electronics')  
SELECT \* FROM Categories

**Output:**

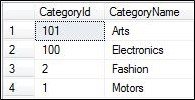


**NOTE :** When IDENTITY\_INSERT is ON, the column list is mandatory in the INSERT Statement.

To turn off IDENTITY\_INSERT, execute the following code:

SET IDENTITY\_INSERT Categories OFF  
INSERT INTO Categories VALUES ('Arts')  
SELECT \* FROM Categories

**Output:**



**Step 5:**

Now, delete the rows from the table Categories and insert a new category Home and observe the value of CategoryId.

DELETE FROM Categories  
INSERT INTO Categories VALUES ('Home')  
SELECT \* FROM Categories

**Output:**



**Step 6:**

Next, execute TRUNCATE query on the Categories table and insert a category Toys and observe the value of CategoryId.

TRUNCATE TABLE Categories  
INSERT INTO Categories VALUES ('Toys')  
SELECT \* FROM Categories

**Output:**



**NOTE :**

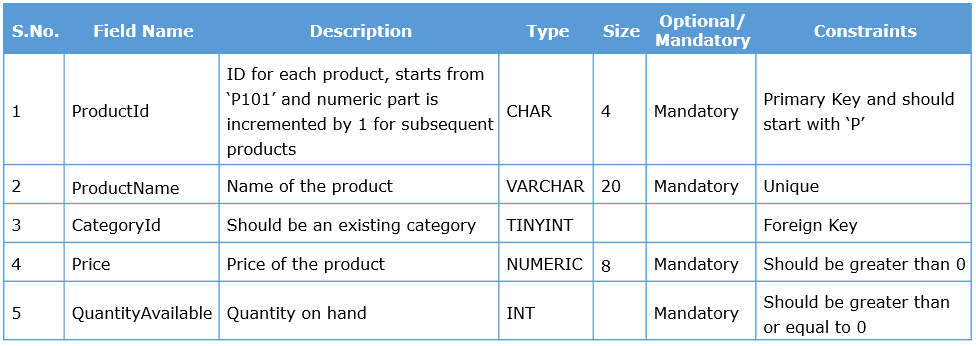
* There can be only one IDENTITY column for a table
* An IDENTITY column does not allow NULL values even though the column is not declared as NOT NULL

**Step 7:**

Now you can create the Products table for the following specification

**Name**:  Products

**Description**:  It is used to store the details of products

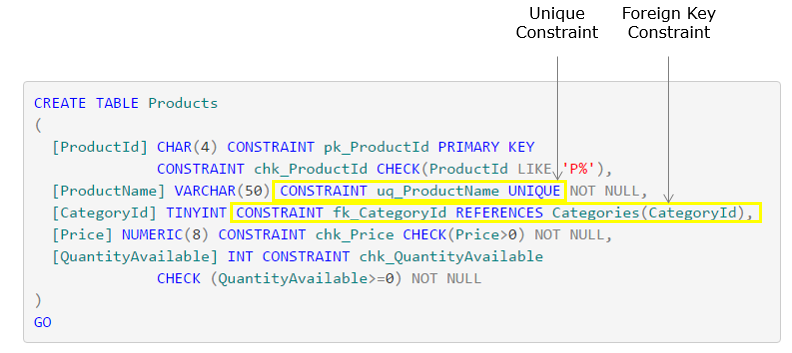


To create the Products table, use the CREATE TABLE statement.

CREATE TABLE Products

**Step 8:**

Add column specifications for Product Id, Product Name, Category to which they belong, Price and Quantity Available, as highlighted.



Download the scripts from this [link](https://academy.onwingspan.com/common-content-store/Shared/Shared/Public/lex_auth_0127560645894225921234_shared/web-hosted/assets/SQL_4.txt) in order to create the table **Products**.

**Step 9:**

Execute the statements and observe the output. Thus, you have achieved the requirement of auto-generating IDs by using IDENTITY.

**NOTE: GO** keyword signals the end of a batch.

Download the scripts from [this](https://academy.onwingspan.com/common-content-store/Shared/Shared/Public/lex_auth_0127560645894225921234_shared/web-hosted/assets/SQL_1.txt) link in order to insert records into the tables **Categories**and **Products**.