* (#1){Screen shot} In a script, create a database and the following tables. Be sure to use the appropriate language so all three statements run with a single click of the Execute button. In the screen shot, display the SQL code and show the Object Explorer AFTER the database and tables are created.
  1. Create a database named *LastName.*Test4 using appropriate DDL statements, where *LastName* is your last name.
  2. Create the tables and appropriate **constraints** based on the following ER diagram.  Use appropriate data types and sizes (see Chapter 8).  The database is for a family-owned bicycle store.
     1. The AvailableOnline column should only accept the letters Y, or N (for Yes or No).
     2. The Cost column should only have values greater than zero.
     3. The SalePrice column should only have values greater than zero.
     4. All columns in both tables are required.
     5. The CID and PID columns should have the IDENTITY property.

|  |  |  |
| --- | --- | --- |
| **Category** |  | **Product** |
| CID | **1                      ∞** | PID |
| Description |  | Description |
|  |  | CID (fk) |
|  |  | Cost  SalePrice |
|  |  | AvailableOnline (Y, N) |

* (#2){2 Screen shots} Add records to the Category table and the Products table. One screenshot should show the SQL code and the output of a SELECT statement showing the records in the table (two screenshots, one for each table) :
  + 1. Insert 3 rows in the Category table:
       1. Bicycles
       2. Components
       3. Accessories
    2. Insert the following records into the Product table:
       1. Schwinn Beach Cruiser, Bicycles category, $79.99, $159.99, N
       2. Diamondback Mountain Bike , Bicycles category, $1,449.99, $2,799.99, N
       3. Cannondale Touring Bicycle, Biclcyes category, $299.00, $625.00, N
       4. Shimano Disc Brake, Components category, $69.89, $149.99, Y
       5. Shimano Derailleur, Components category, $117.88, $229.99, Y
       6. Nitto Mustache Alloy Bar, Components category, $74.99, $94.99, N
       7. Crank Brothers Gem Pump, Accessories category, $12.79, $19.99, Y
       8. Shimano Pro Torque Wrench, Accessories category, $57.99, $139.99, Y
       9. Thule Passage 2 Trunk Rack, Accessories category, $126.95, $149.95, N

* + (#3a){Screenshot.} Create a VIEW named ProductProfit that displays the Category Description, Product Description, and the Profit (SalePrice - Cost). Name the Category description column ‘Category’ and the Product description ‘Item’. The screenshot should show the SQL code to create the view and a SELECT statement displaying the records returned in descending order of the Profit column.
  + (#3b){Screenshot.} Using the ProductProfit VIEW, display the records that are in the TOP 10 Percent of Profit. The screenshot should show the SELECT statement and the records returned.
  + (#4a){Screen shot.} Create a VIEW named FamilyRide that displays the Category description, the Product Description, the Cost and the SalePrice. Name the Category description column ‘Category’ and the Product description ‘Item’. The screenshot should show the SQL code to create the view and a SELECT statement displaying the records returned in ascending order of the Item column.

* (#4b){Screen shot.} Use the FamilyRide VIEW to update the SalePrice of all components to reflect a price increase of 5%. The screenshot should show the SQL code and a SELECT statement displaying the updated records returned in descending order of the SalePrice.
* (#5){Screen Shot} Edit the script so it will execute. Display the SQL code and the output.

USE *LastName*.Test4;

\_\_\_\_\_\_\_\_\_\_\_ @ProductCount INT;

SELECT \_\_\_\_\_\_\_\_\_\_\_\_\_\_ = COUNT(\*)

FROM Product;

IF @ProductCount >= 5

  PRINT 'The number of products is greater than or equal to 5';

\_\_\_\_\_\_\_\_\_

  PRINT 'The number of products is less than 5';

* (#6){Screen shot} Edit the script so it will execute. Display the SQL code and the output.

  USE *LastName*.Test4;

BEGIN \_\_\_\_\_\_\_\_\_

   INSERT Category (\_\_\_\_\_\_\_\_\_\_\_\_\_)

   VALUES ('Clothing');

   PRINT 'SUCCESS: Record was inserted.';

END TRY

BEGIN CATCH

   PRINT 'FAILURE: Record was not inserted.';

   PRINT 'Error ' + \_\_\_\_\_\_\_\_\_\_\_(VARCHAR, ERROR\_NUMBER(), 1)

        + ': ' + ERROR\_MESSAGE();

END \_\_\_\_\_\_\_\_\_\_\_\_;

* (#7){Screen shot} Write a script that removes the view, the two tables, and the database, with one click of the execute button. In the screen shot, display all the SQL code, and show the Object Explorer after the script executes.