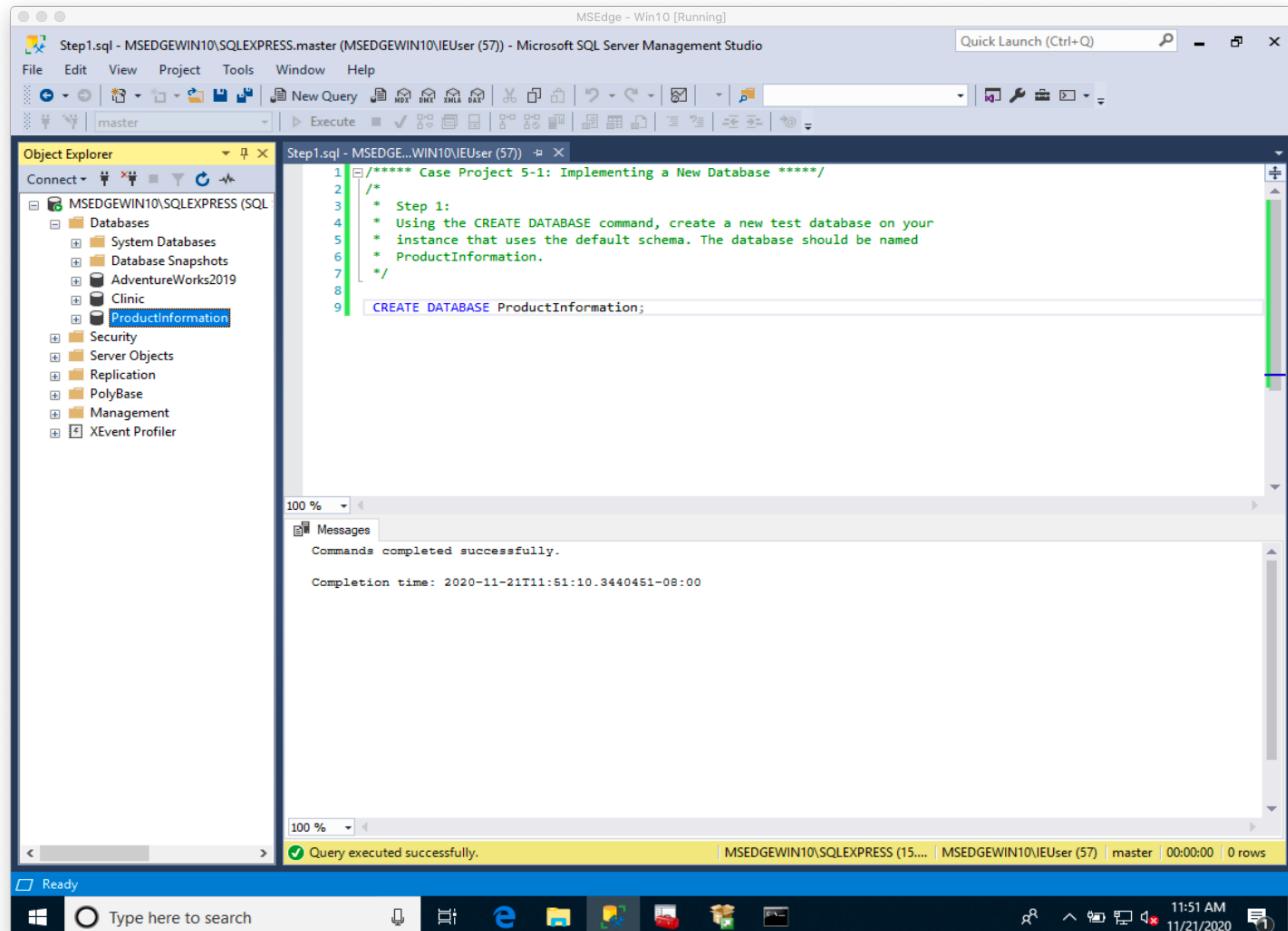


Case Project 5-1: Implementing a New Database

Read case project carefully and submit screen shots proving that you have successfully carried out 1 through 12 successfully.

Each step's queries that I created can be found in my repository at <https://github.com/amajor/DatabaseImplementationLab>

1: CREATE DATABASE



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "Step1.sql - MSEDGEWIN10\SQLEXPRESS.master (MSEDGEWIN10\IEUser (57)) - Microsoft SQL Server Management Studio". The Object Explorer sidebar shows a connection to "MSEDGEWIN10\SQLEXPRESS (SQL Server)" with "Databases" expanded, showing "System Databases", "Database Snapshots", "AdventureWorks2019", "Clinic", and "ProductInformation". The main query editor window contains the following SQL script:

```
1 /* **** Case Project 5-1: Implementing a New Database ****/
2 /*
3 * Step 1:
4 * Using the CREATE DATABASE command, create a new test database on your
5 * instance that uses the default schema. The database should be named
6 * ProductInformation.
7 */
8
9 CREATE DATABASE ProductInformation;
```

The "Messages" pane at the bottom left shows the output: "Commands completed successfully." and "Completion time: 2020-11-21T11:51:10.3440451-08:00". The status bar at the bottom right shows "Ready", "Type here to search", and the system tray with icons for battery, signal, and date/time (11:51 AM, 11/21/2020).

2: ADD TABLES

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the 'ProductInformation' database and its tables: Product, ProductCategory, Supplier, and Inventory. The central pane displays the T-SQL code for creating these four tables. The code uses the 'USE ProductInformation;' command and defines each table with its columns and constraints. The 'Messages' pane at the bottom indicates that the commands completed successfully. The status bar at the bottom right shows the date and time as 11/21/2020 12:31 PM.

```
USE ProductInformation;
CREATE TABLE Product (
    ProductID int NOT NULL,
    ProductCategoryID int NOT NULL,
    SupplierID int NOT NULL,
    Description nvarchar(50) NOT NULL,
    Color nvarchar(20) NOT NULL,
    UnitPrice money NOT NULL,
    ModifiedOn datetime NOT NULL DEFAULT GetDate()
);
CREATE TABLE ProductCategory (
    ProductCategoryID int NOT NULL,
    Description nvarchar(50) NOT NULL,
    ModifiedOn datetime NOT NULL DEFAULT GetDate()
);
CREATE TABLE Supplier (
    SupplierID int,
    SupplierName nvarchar(50) NOT NULL,
    City nvarchar(50) NOT NULL,
    Country nvarchar(50) NOT NULL,
    ContactPhone nvarchar(20) NOT NULL,
    ModifiedOn datetime NOT NULL DEFAULT GetDate()
);
CREATE TABLE Inventory (
    InventoryID int NOT NULL,
    ProductID int NOT NULL,
    UnitsOnHand int NOT NULL,
    ModifiedOn datetime NOT NULL DEFAULT GetDate()
);
```

3: ALTER TABLES

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left displays the database schema, including tables like Product, ProductCategory, and Supplier, along with their constraints, triggers, indexes, and statistics. The central pane contains a script named Step3.sql with the following T-SQL code:

```
5 USE ProductInformation;
6
7 ALTER TABLE Product
8 ADD PRIMARY KEY (ProductID);
9
10 ALTER TABLE ProductCategory
11 ADD PRIMARY KEY (ProductCategoryID);
12
13 ALTER TABLE Supplier
14 ADD PRIMARY KEY (SupplierID);
15
16 ALTER TABLE Inventory
17 ADD PRIMARY KEY (InventoryID);
18
19 ALTER TABLE Product
20 ADD FOREIGN KEY (ProductCategoryID) REFERENCES ProductCategory(ProductCategoryID);
21
22 ALTER TABLE Product
23 ADD FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID);
24
25 ALTER TABLE Inventory
26 ADD FOREIGN KEY (ProductID) REFERENCES Product(ProductID);
```

The status bar at the bottom right indicates the command completed successfully at 12:45 PM on 11/21/2020.

4: INSERT sample data

Supplier Table

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the 'Step4.sql' script window contains T-SQL code for inserting data into the 'Supplier' table. The right side shows the results of a 'SELECT *' query on the 'Supplier' table, displaying five rows of data.

```

USE ProductInformation;
INSERT INTO Supplier (
    SupplierID,
    SupplierName,
    City,
    Country,
    ContactPhone
)
VALUES (
    1,          -- SupplierID
    'Sanzone',  -- SupplierName
    'Oakland',   -- City
    'USA',       -- Country
    '(510) 555-7200' -- ContactPhone
),
(
    2,          -- SupplierID
    'Itex',     -- SupplierName
    'Frankfurt', -- City
    'Germany',   -- Country
    '+49 (0) 8731-9140' -- ContactPhone
),
(
    3,          -- SupplierID
    'Newmix',   -- SupplierName
    'New York',  -- City
    'USA',       -- Country
    '(212) 555-8100' -- ContactPhone
),
(
    4,          -- SupplierID
    'Zenice',   -- SupplierName
    'Chicago',   -- City
    'USA',       -- Country
    '(707) 555-1400' -- ContactPhone
),
(
    5,          -- SupplierID
    'Waredom',  -- SupplierName
    'Paris',     -- City
    'France',   -- Country
    '+33 (0) 123-4496' -- ContactPhone
);
  
```

SupplierID	SupplierName	City	Country	ContactPhone	ModifiedOn
1	Sanzone	Oakland	USA	(510) 555-7200	2020-11-21 12:54
2	Itex	Frankfurt	Germany	+49 (0) 8731-9140	2020-11-21 12:54
3	Newmix	New York	USA	(212) 555-8100	2020-11-21 12:54
4	Zenice	Chicago	USA	(707) 555-1400	2020-11-21 12:54
5	Waredom	Paris	France	+33 (0) 123-4496	2020-11-21 12:54

ProductCategory Table

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the 'Step4.sql' script window contains T-SQL code for inserting data into the 'ProductCategory' table. The right side shows the results of a 'SELECT *' query on the 'ProductCategory' table, displaying five rows of data.

```

INSERT INTO ProductCategory (
    ProductCategoryID,
    Description
)
VALUES (
    1,          -- ProductCategoryID
    'Cookware' -- Description
),
(
    2,          -- ProductCategoryID
    'Cutlery'  -- Description
),
(
    3,          -- ProductCategoryID
    'Linens'   -- Description
),
(
    4,          -- ProductCategoryID
    'Tableware' -- Description
),
(
    5,          -- ProductCategoryID
    'Cooks Tools' -- Description
);
  
```

ProductCategoryID	Description	ModifiedOn
1	Cookware	2020-11-21 13:00:05.403
2	Cutlery	2020-11-21 13:00:05.403
3	Linens	2020-11-21 13:00:05.403
4	Tableware	2020-11-21 13:00:05.403
5	Cooks Tools	2020-11-21 13:00:05.403

Product Table

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the 'Step4.sql' script window contains an INSERT INTO statement for the 'Product' table. The right side shows the 'SQLQuery5.sql' window with a SELECT * query and the results grid.

```

68: INSERT INTO Product (
69:     ProductID,
70:     ProductCategoryID,
71:     SupplierID,
72:     Description,
73:     Color,
74:     UnitPrice
75: )
76: VALUES (
77:     1, -- ProductID
78:     2, -- ProductCategoryID
79:     3, -- SupplierID
80:     'Stainless steel flatware', -- Description
81:     'Silver', -- Color
82:     30.99 -- UnitPrice
83: ),
84: (
85:     2, 2, 4,
86:     'Serving spoons',
87:     'Nickle',
88:     18.5
89: ),
90: (
91:     3, 4, 1,
92:     'Chef''s knife',
93:     'Silver',
94:     25
95: ),
96: (
97:     . . .

```

Results Grid:

ProductID	ProductCategoryID	SupplierID	Description	Color	UnitPrice	ModifiedOn
1	2	3	Stainless steel flatware	Silver	30.99	2020-11-21 11:53:183
2	2	4	Serving spoons	Nickle	18.50	2020-11-21 11:53:183
3	3	4	Chef's knife	Silver	25.00	2020-11-21 11:53:183
4	4	5	Cutting board	Black	15.77	2020-11-21 11:53:183
5	5	2	Sharpening steel	Carbon	12.24	2020-11-21 11:53:183
6	6	3	Napkin set	Red	9.31	2020-11-21 11:53:183
7	7	3	Table cloth	Various	21.89	2020-11-21 11:53:183
8	8	1	Large frying pan	Black	13.50	2020-11-21 11:53:183
9	9	3	Small frying pan	Black	7.25	2020-11-21 11:53:183
10	10	3	Nonstick saucepan	Silver	16.00	2020-11-21 11:53:183

Inventory Table

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the 'Step4.sql' script window contains an INSERT INTO statement for the 'Inventory' table. The right side shows the 'SQLQuery5.sql' window with a SELECT * query and the results grid.

```

130: INSERT INTO Inventory (
131:     InventoryID,
132:     ProductID,
133:     UnitsOnHand
134: )
135: VALUES (
136:     1, -- InventoryID
137:     1, -- ProductID
138:     40 -- UnitsOnHand
139: ),
140: (
141:     2, 2, 24
142: ),
143: (
144:     3, 4, 5
145: ),
146: (
147:     4, 5, 2
148: ),
149: (
150:     5, 6, 7
151: ),
152: (
153:     6, 7, 16
154: ),
155: (
156:     7, 8, 12
157: ),
158: (
159:     8, 10, 27
160: );

```

Results Grid:

InventoryID	ProductID	UnitsOnHand	ModifiedOn
1	1	40	2020-11-21 13:11:53.183
2	2	24	2020-11-21 13:11:53.183
3	3	5	2020-11-21 13:11:53.183
4	4	2	2020-11-21 13:11:53.183
5	5	7	2020-11-21 13:11:53.183
6	6	16	2020-11-21 13:11:53.183
7	7	12	2020-11-21 13:11:53.183
8	10	27	2020-11-21 13:11:53.183

5: UPDATE data

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "Step5.sql - MSEDGEWIN10\SQLEXPRESS.ProductInformation (MSEDGEWIN10\IEUser (51)) - Microsoft SQL Server Management...". The main window displays the following SQL script:

```
1 /* **** Case Project 5-1: Implementing a New Database *****/
2 /* Step 5:
3  * Use an UPDATE statement to change the SupplierName of SupplierID 2 to
4  * Cuisinex.
5  *****/
6 USE ProductInformation;
7
8 /* Select only the row we care about to test our condition */
9 SELECT * FROM Supplier WHERE SupplierID = 2;
10
11 /* Perform the update */
12 UPDATE Supplier
13     SET SupplierName = 'Cuisinex'
14     WHERE SupplierID = 2;
15
16 /* See our update in context with the rest of the data */
17 SELECT * FROM Supplier;
```

The results pane shows two tables. The first table has one row:

	SupplierID	SupplierName	City	Country	ContactPhone	ModifiedOn
1	2	Itex	Frankfurt	Germany	+49 (0) 8731-9140	2020-11-21 12:54:41.180

The second table has five rows:

	SupplierID	SupplierName	City	Country	ContactPhone	ModifiedOn
1	1	Sanzone	Oakland	USA	(510) 555-7200	2020-11-21 12:54:41.180
2	2	Cuisinex	Frankfurt	Germany	+49 (0) 8731-9140	2020-11-21 12:54:41.180
3	3	Newnix	New York	USA	(212) 555-8100	2020-11-21 12:54:41.180
4	4	Zenice	Chicago	USA	(707) 555-1400	2020-11-21 12:54:41.180
5	5	Waredom	Paris	France	+33 (0) 123-4496	2020-11-21 12:54:41.180

At the bottom, a message bar indicates "Query executed successfully." and shows the session details: MSEDGEWIN10\SQLEXPRESS (15..., MSEDGEWIN10\IEUser (51), ProductInformation, 00:00:00, 5 rows). The taskbar at the bottom right shows the date and time as 11/21/2020 1:19 PM.

6: DELETE data

Here are our queries:

```

USE ProductInformation;
/* Test our condition to confirm the data we'll remove */
SELECT * FROM Product WHERE Description = 'Nonstick saucepan'; -- This returns ProductID = 10
SELECT * FROM Inventory; -- We will want the record where ProductID = 10, InventoryID = 8
/* See the products with removed data in context */
SELECT * FROM Inventory
WHERE ProductID IN (
    SELECT ProductID FROM Product WHERE Description = 'Nonstick saucepan'
);
/* Remove the data */
DELETE FROM Inventory
WHERE ProductID IN (
    SELECT ProductID FROM Product WHERE Description = 'Nonstick saucepan'
);
/* Confirm inventory for ProductID = 10, InventoryID = 8 is now gone */
SELECT * FROM Inventory; -- We will want the record where ProductID = 10

```

And the output:

ProductID	ProductCategoryID	SupplierID	Description	Color	UnitPrice	ModifiedOn
1	1	3	Nonstick saucepan	Silver	16.00	2020-11-21 13:08:20.000

InventoryID	ProductID	UnitsOnHand	ModifiedOn
1	1	40	2020-11-21 13:11:53.183
2	2	24	2020-11-21 13:11:53.183
3	4	5	2020-11-21 13:11:53.183
4	5	2	2020-11-21 13:11:53.183
5	6	7	2020-11-21 13:11:53.183
6	7	16	2020-11-21 13:11:53.183
7	8	12	2020-11-21 13:11:53.183
8	10	27	2020-11-21 13:11:53.183

7: LEFT OUTER JOIN

The screenshot shows the Microsoft SQL Server Management Studio interface running on Windows 10. The title bar reads "MSEdge - Win10 [Running]". The query window displays the following SQL code:

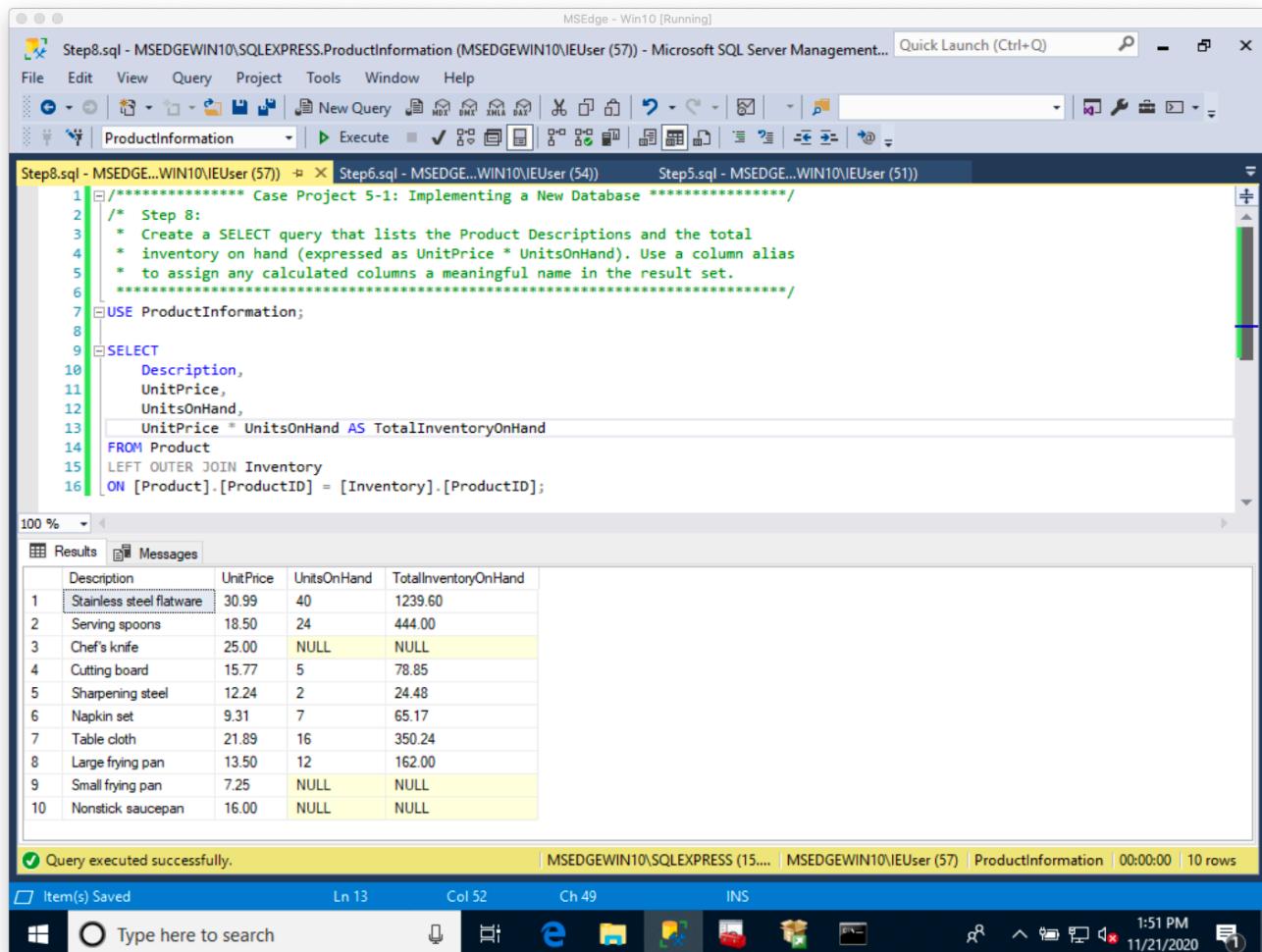
```
Step7.sql - MSEDGEWIN10\SQLEXPRESS.ProductInformation (MSEDGEWIN10\IEUser (57)) - Microsoft SQL Server Management...
File Edit View Query Project Tools Window Help
ProductInformation Execute
Step7.sql - MSEDGE...WIN10\IEUser (57) Step6.sql - MSEDGE...WIN10\IEUser (54) Step5.sql - MSEDGE...WIN10\IEUser (51)
1 /* **** Case Project 5-1: Implementing a New Database *****/
2 /* Step 7:
3  * Using a LEFT OUTER JOIN, retrieve all Product Descriptions that have no
4  * associated inventory.
5  * **** */
6 USE ProductInformation;
7
8 SELECT Description
9 FROM Product
10 LEFT OUTER JOIN Inventory
11 ON [Product].[ProductID] = [Inventory].[ProductID]
12 WHERE InventoryID IS NULL;
```

The results pane shows a table with one column "Description" containing three rows:

Description
Chef's knife
Small frying pan
Nonstick saucepan

At the bottom of the interface, the status bar indicates "Query executed successfully." and "3 rows". The taskbar at the bottom of the screen shows the Windows Start button, a search bar, and several pinned application icons.

8: SELECT total inventory on hand



The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "MSEdge - Win10 [Running]". The query editor window contains a SQL script named "Step8.sql" with the following code:

```
1 /* **** Case Project 5-1: Implementing a New Database *****/
2 /* Step 8:
3  * Create a SELECT query that lists the Product Descriptions and the total
4  * inventory on hand (expressed as UnitPrice * UnitsOnHand). Use a column alias
5  * to assign any calculated columns a meaningful name in the result set.
6  * *****/
7 USE ProductInformation;
8
9 SELECT
10    Description,
11    UnitPrice,
12    UnitsOnHand,
13    UnitPrice * UnitsOnHand AS TotalInventoryOnHand
14 FROM Product
15 LEFT OUTER JOIN Inventory
16    ON [Product].[ProductID] = [Inventory].[ProductID];
```

The results grid displays 10 rows of data:

	Description	UnitPrice	UnitsOnHand	TotalInventoryOnHand
1	Stainless steel flatware	30.99	40	1239.60
2	Serving spoons	18.50	24	444.00
3	Chef's knife	25.00	NULL	NULL
4	Cutting board	15.77	5	78.85
5	Sharpening steel	12.24	2	24.48
6	Napkin set	9.31	7	65.17
7	Table cloth	21.89	16	350.24
8	Large frying pan	13.50	12	162.00
9	Small frying pan	7.25	NULL	NULL
10	Nonstick saucepan	16.00	NULL	NULL

At the bottom of the screen, the taskbar shows the following icons and information:

- Windows Start button
- Type here to search input field
- File Explorer icon
- Recycle Bin icon
- Task View icon
- Power icon
- Volume icon
- Network icon
- 1:51 PM
- 11/21/2020

9: SELECT top two products with most inventory

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'Step9.sql - MSEdge...WIN10\IEUser (51)' is open, displaying the following SQL code:

```
1 /* **** Case Project 5-1: Implementing a New Database *****/
2 /* Step 9:
3  * Create a SELECT statement that returns the top two products with the most
4  * inventory units on hand.
5  * **** */
6 USE ProductInformation;
7
8 SELECT TOP (2)
9     Product.ProductID,
10    ProductDescription,
11    Color,
12    UnitPrice,
13    UnitsOnHand
14 FROM Product
15 LEFT OUTER JOIN Inventory
16 ON [Product].[ProductID] = [Inventory].[ProductID]
17 ORDER BY UnitsOnHand DESC;
```

The results pane shows a table with two rows of data:

	ProductID	Description	Color	UnitPrice	UnitsOnHand
1	1	Stainless steel flatware	Silver	30.99	40
2	2	Serving spoons	Nickle	18.50	24

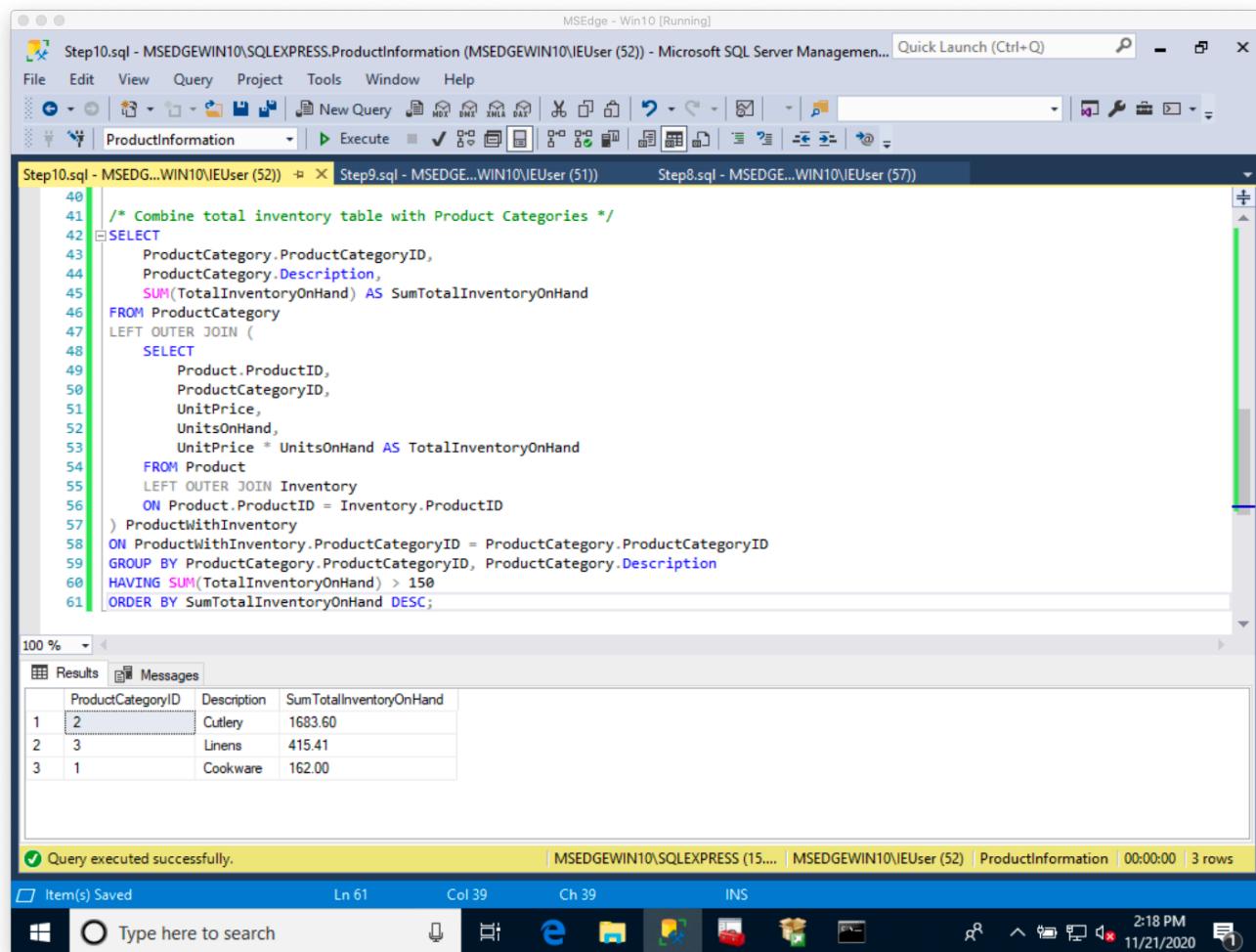
At the bottom of the interface, a status bar indicates: 'Query executed successfully.' and 'MSEdgeWIN10\SQLEXPRESS (15... | MSEdgeWIN10\IEUser (51) | ProductInformation | 00:00:00 | 2 rows'.

10: SELECT total inventory on hand by category greater than \$150

We can calculate our total inventory on hand using a join between Product and Inventory tables.

	ProductID	ProductCategoryID	UnitPrice	UnitsOnHand	TotalInventoryOnHand
1	8	1	13.50	12	162.00
2	9	1	7.25	NULL	NULL Category Total: 162.00
3	10	1	16.00	NULL	NULL \$162.00
4	1	2	30.99	40	1239.60 Category Total: 1239.60
5	2	2	18.50	24	444.00 \$1683.60
6	6	3	9.31	7	65.17 Category Total: 65.17
7	7	3	21.89	16	350.24 \$415.41
8	3	4	25.00	NULL	NULL
9	4	5	15.77	5	78.85 Category Total: 78.85
10	5	5	12.24	2	24.48 \$103.33

We can use this information to double check our final result where we use GROUP BY to add together total inventory and return only the sum total inventories for categories that are over \$150:



```

40  /* Combine total inventory table with Product Categories */
41  SELECT
42      ProductCategory.ProductCategoryID,
43      ProductCategory.Description,
44      SUM(TotalInventoryOnHand) AS SumTotalInventoryOnHand
45  FROM ProductCategory
46  LEFT OUTER JOIN (
47      SELECT
48          Product.ProductID,
49          ProductCategoryID,
50          UnitPrice,
51          UnitsOnHand,
52          UnitPrice * UnitsOnHand AS TotalInventoryOnHand
53      FROM Product
54      LEFT OUTER JOIN Inventory
55      ON Product.ProductID = Inventory.ProductID
56  ) ProductWithInventory
57  ON ProductWithInventory.ProductCategoryID = ProductCategory.ProductCategoryID
58  GROUP BY ProductCategory.ProductCategoryID, ProductCategory.Description
59  HAVING SUM(TotalInventoryOnHand) > 150
60  ORDER BY SumTotalInventoryOnHand DESC;
61

```

The screenshot shows the SSMS interface with the following details:

- Query Window:** Step10.sql - MSEDGEWIN10\SQLEXPRESS.ProductInformation (MSEDGEWIN10\IEUser (52)) - Microsoft SQL Server Management... (Step10.sql)
- Results Grid:** Shows the output of the query, which includes three rows of data:

ProductCategoryID	Description	SumTotalInventoryOnHand
2	Cutlery	1683.60
3	Linens	415.41
1	Cookware	162.00
- Status Bar:** Query executed successfully. | MSEDGEWIN10\SQLEXPRESS (15...) | MSEDGEWIN10\IEUser (52) | ProductInformation | 00:00:00 | 3 rows
- Taskbar:** Item(s) Saved, Ln 61, Col 39, Ch 39, INS, Type here to search, 2:18 PM, 11/21/2020

11: Create View

```

1 /* **** Case Project 5-1: Implementing a New Database **** */
2 /* Step 11:
3  * Create a view vTotalInventoryByProductCategory that lists the
4  * ProductCategory description and the total inventory on hand.
5  */
6 CREATE VIEW vTotalInventoryByProductCategory AS
7     SELECT
8         ProductCategory.ProductCategoryID AS 'Category ID',
9         ProductCategory.Description AS 'Description',
10        SUM(TotalInventoryOnHand) AS 'Total Inventory On Hand'
11    FROM ProductCategory
12    LEFT OUTER JOIN (
13        SELECT
14            Product.ProductID,
15            ProductCategoryID,
16            UnitPrice,
17            UnitsOnHand,
18            UnitPrice * UnitsOnHand AS TotalInventoryOnHand
19        FROM Product
20        LEFT OUTER JOIN Inventory
21            ON Product.ProductID = Inventory.ProductID
22    ) ProductWithInventory
23    ON ProductWithInventory.ProductCategoryID = ProductCategory.ProductCategoryID
24    GROUP BY ProductCategory.ProductCategoryID, ProductCategory.Description

```

Messages

Commands completed successfully.

Completion time: 2020-11-21T14:37:45.8443430-08:00

Query executed successfully | MSEDGEWIN10\SQLEXPRESS (15...) | MSEDGEWIN10\IEUser (54) | ProductInformation | 00:00:00 | 0 rows

Show the data found in the new view:

```

1 /* ****Show VIEW Data ****/
2 SELECT *
3     FROM [ProductInformation].[dbo].[vTotalInventoryByProductCategory]
4     ORDER BY [Total Inventory On Hand] DESC

```

	Category ID	Description	Total Inventory On Hand
1	2	Cutlery	1683.60
2	3	Linens	415.41
3	1	Cookware	162.00
4	5	Cooks Tools	103.33
5	4	Tableware	NULL

12: Add Index

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows the database structure for 'ProductInformation'. The 'Tables' node is expanded, showing 'dbo.Product' with its columns, keys, constraints, triggers, and indexes. One index, 'IX_Product_Description', is highlighted. The 'Indexes' node also lists 'PK_Product_B40CC6ED6A9494CA' (Clustered). The 'Messages' pane at the bottom right shows the command completed successfully.

```
1 /* **** Case Project 5-1: Implementing a New Database ****/
2 /* Step 12:
3 *   * Add an index to the Product table on the Description column.
4 * **** */
5 USE ProductInformation;
6
7 CREATE INDEX IX_Product_Description
8 ON Product (
9     Description
10 );
```

Commands completed successfully.
Completion time: 2020-11-21T14:43:15.5181185-08:00