

KillTest

Higher Quality, Better Service!



Q&A

<http://www.killtest.com>

We offer free update service for one year.

Exam : 070-761

**Title : Querying Data with
Transact-SQL**

Version : Demo

1.You create a table named Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
    ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    ProductName nvarchar(100) NULL,
    UnitPrice decimal(18, 2) NOT NULL,
    UnitsInStock int NOT NULL,
    UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
    VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    SET XACT_ABORT ON
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
            VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF XACT_STATE() <> 0 ROLLBACK TRANSACTION
        THROW 51000, 'The product could not be created.', 1
    END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

2.You create a table named Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
    ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    ProductName nvarchar(100) NULL,
    UnitPrice decimal(18, 2) NOT NULL,
    UnitsInStock int NOT NULL,
    UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```
CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
    VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
END
```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```
ALTER PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    SET XACT_ABORT ON
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products(ProductName,ProductPrice,ProductsInStock,ProductsOnOrder)
            VALUES (@ProductName,@UnitPrice,@UnitsInStock,@UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF XACT_STATE() <> 0 ROLLBACK TRANSACTION
        THROW 51000, 'The product could not be created.', 1
    END CATCH
END
```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

3.You create a table named Products by running the following Transact-SQL statement:

```
CREATE TABLE Products (
    ProductID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    ProductName nvarchar(100) NULL,
    UnitPrice decimal(18, 2) NOT NULL,
    UnitsInStock int NOT NULL,
    UnitsOnOrder int NULL
)
```

You have the following stored procedure:

```

CREATE PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    INSERT INTO Products (ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
    VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
END

```

You need to modify the stored procedure to meet the following new requirements:

- Insert product records as a single unit of work.
- Return error number 51000 when a product fails to insert into the database.
- If a product record insert operation fails, the product information must not be permanently written to the database.

Solution: You run the following Transact-SQL statement:

```

ALTER PROCEDURE InsertProduct
    @ProductName nvarchar(100),
    @UnitPrice decimal(18,2),
    @UnitsInStock int,
    @UnitsOnOrder int
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO Products (ProductName, ProductPrice, ProductsInStock, ProductsOnOrder)
            VALUES (@ProductName, @UnitPrice, @UnitsInStock, @UnitsOnOrder)
        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION
        IF @@ERROR = 51000
            THROW
    END CATCH
END

```

Does the solution meet the goal?

- A. Yes
- B. No

Answer: A

4. You create a table named Customer by running the following Transact-SQL statement:

```

CREATE TABLE Customer (
    CustomerID int IDENTITY(1,1) PRIMARY KEY,
    FirstName varchar(50) NULL,
    LastName varchar(50) NOT NULL,
    DateOfBirth date NOT NULL,
    CreditLimit money CHECK (CreditLimit < 10000),
    TownID int NULL REFERENCES dbo.Town(TownID),
    CreatedDate datetime DEFAULT(Getdate())
)

```

You must insert the following data into the Customer table:

Record	First name	Last name	Date of Birth	Credit limit	Town ID	Created date
Record 1	Yvonne	McKay	1984-05-25	9,000	no town details	current date and time
Record 2	Jossef	Goldberg	1995-06-03	5,500	no town details	current date and time

You need to ensure that both records are inserted or neither record is inserted.

Solution: You run the following Transact-SQL statement:

```
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000, GETDATE())
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500, GETDATE())
GO
```

Does the solution meet the goal?

A. Yes

B. No

Answer: B

5. You create a table named Customer by running the following Transact-SQL statement:

```
CREATE TABLE Customer (
    CustomerID int IDENTITY(1,1) PRIMARY KEY,
    FirstName varchar(50) NULL,
    LastName varchar(50) NOT NULL,
    DateOfBirth date NOT NULL,
    CreditLimit money CHECK (CreditLimit < 10000),
    TownID int NULL REFERENCES dbo.Town(TownID),
    CreatedDate datetime DEFAULT(Getdate())
)
```

You must insert the following data into the Customer table:

Record	First name	Last name	Date of Birth	Credit limit	Town ID	Created date
Record 1	Yvonne	McKay	1984-05-25	9,000	no town details	current date and time
Record 2	Jossef	Goldberg	1995-06-03	5,500	no town details	current date and time

You need to ensure that both records are inserted or neither record is inserted.

Solution: You run the following Transact-SQL statement:

```
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Yvonne', 'McKay', '1984-05-25', 9000, GETDATE())
INSERT INTO Customer (FirstName, LastName, DateOfBirth, CreditLimit, CreatedDate)
VALUES ('Jossef', 'Goldberg', '1995-06-03', 5500, GETDATE())
GO
```

Does the solution meet the goal?

A. Yes

B. No

Answer: B