

70-762^{Q&As}

Developing SQL Databases

Pass Microsoft 70-762 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

<https://www.pass2lead.com/70-762.html>

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Microsoft
Official Exam Center

-  **Instant Download** After Purchase
-  **100% Money Back** Guarantee
-  **365 Days** Free Update
-  **800,000+** Satisfied Customers



QUESTION 1

Note: This question is part of a series of questions that use the same answer choices. An answer choice may be correct for more than one question on the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.

You work on an OLTP database that has no memory-optimized file group defined.

You have a table names tblTransaction that is persisted on disk and contains the information described in the following table:

Item	Name	Data Type	Nullable	Notes
Column	TransactionDate	Date	No	For each transaction date, there are only about 100,000 records. The table contains over one billion records in total.
Column	SequenceNo	bigint	No	Uniquely identifies a transaction record within a date
Column	AccountId	int	No	
Column	ValueType	char(3)	No	
Column	Amount	decimail(20.2)	Yes	
	IX_ValueType			Nonclustered columnstore index on the ValueType column.

Users report that the following query takes a long time to complete.

```
SELECT TransactionDate, COUNT(*) AS TotalCount FROM tblTransaction
WHERE TransactionDate - DATEADD(D, -1, CONVERT (DATE, CONVERT (VARCHAR (8),
GETDATE (), 112) 112))
GROUP BY TransactionDate;
```

You need to create an index that:

- improves the query performance
- does not impact the existing index
- minimizes storage size of the table (inclusive of index pages).

What should you do?

Users report that the following query takes a long time to complete.

```
SELECT TransactionDate, COUNT(*) AS TotalCount FROM tblTransaction
WHERE TransactionDate - DATEADD(D, -1, CONVERT (DATE, CONVERT (VARCHAR (8),
GETDATE (), 112) 112))
GROUP BY TransactionDate;
```

You need to create an index that:

- improves the query performance
- does not impact the existing index
- minimizes storage size of the table (inclusive of index pages).

What should you do?

- A. Create a clustered index on the table.
- B. Create a nonclustered index on the table.
- C. Create a nonclustered filtered index on the table.
- D. Create a clustered columnstore index on the table.
- E. Create a nonclustered columnstore index on the table.
- F. Create a hashindex on the table.

Correct Answer: C

A filtered index is an optimized nonclustered index, especially suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve query performance, reduce index maintenance costs, and reduce index storage costs compared with full-table indexes.

QUESTION 2

You have the following stored procedure that is called by other stored procedures and applications:

```
CREATE PROCEDURE UpdateCustomer @CustomerId INT
AS
BEGIN
    EXEC ProcessCustomer_Internal @CustomerId
    DECLARE @Status INT
    SELECT @Status = Status FROM Customer WHERE CustomerId =
@CustomerId
END
```

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

Always return a value to the caller.

Return 0 if @Status is NULL.

Callers must be able to use @Status as a variable.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

- A. Replace NULL values with 0. Add a PRINT statement to return @Status.

- B. Add a RETURN statement.
- C. Replace NULL values with 0. Add an output parameter to return @Status.
- D. Replace NULL values with 0. Add a SELECT statement to return @Status.
- E. Add a PRINT statement.
- F. Add a SELECT statement to return @Status.
- G. Add an output parameter to return @Status.

Correct Answer: BC

There are three ways of returning data from a procedure to a calling program: result sets, output parameters, and return codes.

References: <https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/return-data-from-a-stored-procedure?view=sql-server-2017>

QUESTION 3

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution. Determine whether the solution meets the stated goals.

You need to create a stored procedure that updates the Customer, CustomerInfo, OrderHeader, and OrderDetails tables in order.

You need to ensure that the stored procedure:

Runs within a single transaction.

Commits updates to the Customer and CustomerInfo tables regardless of the status of updates to the OrderHeader and OrderDetail tables.

Commits changes to all four tables when updates to all four tables are successful.

Solution: You create a stored procedure that includes the following Transact-SQL segment:

```
DECLARE @CustomerComplete bit = 0
BEGIN TRY
    BEGIN TRAN
        UPDATE Customer ...
        UPDATE CustomerInfo ...

        SET @CustomerComplete = 1

        SAVE TRAN TR1

        UPDATE OrderHeader ...
        UPDATE OrderDetail ...
    COMMIT TRAN
END TRAN
BEGIN CATCH
    IF (@CustomerComplete = 1) AND (XACT_STATE() = 1)
    BEGIN
        COMMIT TRAN
    END
    ELSE IF XACT_STATE() = 1
        ROLLBACK TRAN
END CATCH
```

Does the solution meet the goal?

- A. Yes
- B. No

Correct Answer: A

Need to handle the case where the first two updates (OrderHeader, OrderDetail) are successful, but either the 3rd or the 4th (OrderHeader, OrderDetail) fail. We add the @CustomerComplete variable in the BEGIN TRY block, and test it in the BEGIN CATCH block.

Note: XACT_STATE indicates whether the request has an active user transaction, and whether the transaction is capable of being committed. XACT_STATE =1: the current request has an active user transaction. The request can perform any actions, including writing data and committing the transaction.

References: <https://docs.microsoft.com/en-us/sql/t-sql/functions/xact-state-transact-sql>

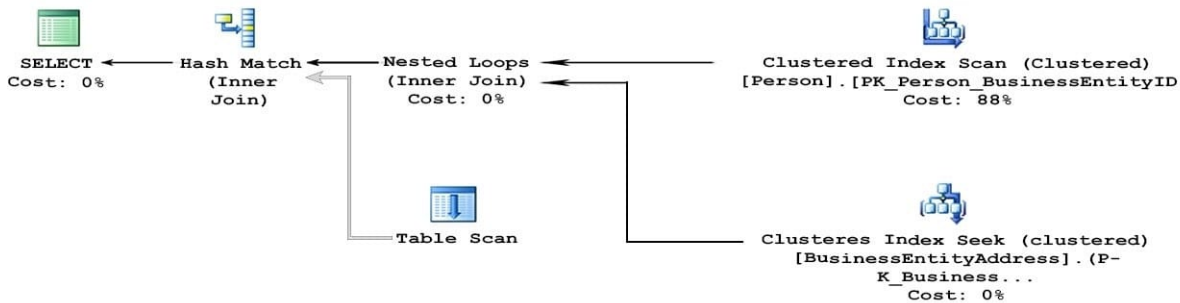
QUESTION 4

Note: This question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in this series. Information and details provided in a question apply only to that question.

You have a database named DB1. There is no memory-optimized filegroup in the database. You run the following query:

```
DECLARE @LastName varchar(20) = "contoso";  
SELECT P.FirstName, Addr.*  
From Person.Person P  
JOIN Person.BusinessEntityAddress On P.BusinessEntityID =  
BusinessEntityAddress.BusinessEntityID  
JOIN Person.[Address] Addr ON BusinessEntityAddress.AddressID =  
Addr.AddressID  
WHERE P.LastName = @LastName;
```

The following image displays the execution plan the query optimizer generates for this query:



Users frequently run the same query with different values for the local variable @lastName. The table named Person is persisted on disk. You need to create an index on the Person.Person table that meets the following requirements:

1. All users must be able to benefit from the index.
2. FirstName must be added to the index as an included column.

What should you do?

- A. Create a clustered index on the table.
- B. Create a nonclustered index on the table.
- C. Create a nonclustered filtered index on the table.
- D. Create a clustered columnstore index on the table.
- E. Create a nonclustered columnstore index on the table.
- F. Create a hash index on the table.

Correct Answer: B

By including nonkey columns, you can create nonclustered indexes that cover more queries. This is because the nonkeycolumns have the following benefits:

They can be data types not allowed as index key columns.

They are not considered by the Database Engine when calculating the number of index key columns or index key size.

QUESTION 5

Note: this question is part of a series of questions that use the same or similar answer choices. An answer choice may be correct for more than one question in the series. Each question is independent of the other questions in the series.

Information and details provided in a question apply only to that question.

You are developing an application to track customer sales.

You need to create an object that meets to following requirements:

Run managed code packaged in an assembly that was created in the Microsoft.NET Framework and uploaded in Microsoft SQL Server.

Run written a transaction and roll back if a failure occurs.

Run when a table is created or modified.

What should you create?

- A. extended procedure
- B. CLR procedure
- C. user-defined procedure
- D. DML trigger
- E. DDL trigger
- F. scalar-valued function
- G. table-valued function

Correct Answer: B

The common language runtime (CLR) is the heart of the Microsoft .NET Framework and provides the execution environment for all .NET Framework code. Code that runs within the CLR is referred to as managed code. With the CLR hosted in Microsoft SQL Server (called CLR integration), you can author stored procedures, triggers, user-defined functions, user-defined types, and user-defined aggregates in managed code. References:

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/introduction-to-sql-server-clr-integration>