**Course Name: .Net 6 Core Entity Framework: How-To Guide For Professionals**

*NOTE: Mark the correct answers with Yellow highlight*

**Chapter Number 4 – CRUD Operations**

1. What basic database operations does the anacronym CRUD represent?

a. Convert, Revise, Undo, and Destroy.

Reason – Incorrect; these are not discrete database operations.

b. Create, Remove, Undo, and Deploy.

Reason – Incorrect, there is no Deploy operations, while Undo is achieved using a rollback operation before the transaction is committed (or discarding the transaction).

c. Create, Read, Undo, and Delete.

Reason – Incorrect, missing the Update (U) operation, while Undo is achieved using a rollback operation before the transaction is committed (or discarding the transaction).

d. Create, Read, Update, and Delete

Reason – Correct, these are the four atomic database operations.

2. The await keyword is used for?

a. To suspend the current process and wait for a database operation to complete.

Reason – Incorrect, it will release the dependence on the database operation so that other application / user operations can complete.

b. to release the current thread in support of asynchronous database operations, typically for long running queries, updates, or stored procedure invocations.

Reason – Correct, it is a method modifier that indicate there is one or more contained asynchronous database operation being performed that free (release) the single thread used by the entity framework to perform operations (thus making the application more performant since it is not held-up waiting for long-running database operations, and will resume processing once the operation completes).

c. To enable multiple long running data operations to be invoked in parallel, simulating a multi-threaded environment.

Reason – Incorrect; while it does enable data operations to run untethered to the application thread, the .net runtime is single threaded by default – other mechanisms are used to achieve parallel processing.

d. To pause the application process and wait for database operation to complete.

Reason – Incorrect; the application switches to perform operations on a different application user-session.

3. How do you ensure you get only one result when you perform a query on a ***DbContext*** derived class?

a. Using synchronous operations Single() and FirstOrDefault() only.

Reason – Incorrect, there are asynchronous implementations for each of these methods that also ensure a single result is returned.

b. Use the synchronous operations Single() or FirstOrDefault(), or asynchronous versions SingleAsync() or FirstOrDefaultAsync().

Reason – Correct, however Single() and SingleAsync() will throw an exception if no results or too many results are found.

c. Use the synchronous operation FirstOnly() or asynchronous operation FirstOnlyAsync().

Reason – Incorrect; there are no such methods.

d. Using the keyword modifier OneOnly() for both asynchronous and synchronous operations.

Reason – Incorrect; there is no such method and, by convention asynchronous operations have their method name suffixed by “Async”. For example “FirstOrDefault” and “FirstOrDefaultAsync”.

4. When accessing data from a primary table in a database, how do you retrieve it children as well (re: a parent-child relationship)?

a. They are included by default in the result set, and accessible via the navigation chain from the parent.

Reason – Incorrect; child data are not included by default in the returned results.

b. By using the Children() method to identify the child collections to be included in the result set.

Reason – Incorrect; there is no method called Children().

c. You cannot access child data as dependents of a parent, they must be queried for independently.

Reason – Incorrect; you can include any dependent children of a parent object by using the Include() method to identify each child data set.

d. Use the Include() method to identify each child entity collection to be included in the result set.

Reason – Correct; the Include method is used to specify the related entities to be included in the result set and mapped to the corresponding related class.

5. To complete the Inserting, Updating, or Deleting data operation, you must…

a. remember to call Finalize() on the data operation.

Reason – Incorrect; there is no such method.

b. always call CommitDb() on the DbContext, to commit all changes to the back-end database.

Reason – Incorrect; while databases do use the commit sql command to complete a transaction, this is performed using the SaveChanges() method of the DbContext .net entity framework object.

c. to call SaveChanges() on the DbContext, to commit all changes to the back-end database.

Reason – Correct, all data operations require the database transaction to be committed and made available to all other database users, and this is achieved by calling SaveChanges on the DbContext instance.

d. You must use the IQueryable interface for all persistence operations.

Reason –Incorrect; IQueryable interface is used to retrieve data from the database, not perform updates (or delete).