1. **Entity Framework**

Entity Framework is an Object Relational Mapper (ORM) which is a type of tool that simplifies mapping between objects in your software to the tables and columns of a relational database.

* 1. **Types:**

POCO - just auto create in visual studio - A POCO entity is a class that doesn't depend on any framework-specific base class. It is like any other normal .NET CLR class, which is why it is called "Plain Old CLR Objects".

Dynamic – change some properties attribute based

* 1. **Entity Lifecycle**

States in lifetime –

* Added: The entity is marked as added.
* Deleted: The entity is marked as deleted.
* Modified: The entity has been modified.
* Unchanged: The entity hasn’t been modified.
* Detached: The entity isn’t tracked.
  1. **State:**
* **Unchanged State**
* When an entity is Unchanged, it’s bound to the context but it hasn’t been modified.
* By default, an entity retrieved from the database is in this state.
* When an entity is attached to the context (with the Attach method), it similarly is in the Unchanged state.
* The context can’t track changes to objects that it doesn’t reference, so when they’re attached it assumes they’re Unchanged.
* **Detached State**
* Detached is the default state of a newly created entity because the context can’t track the creation of any object in your code.
* This is true even if you instantiate the entity inside a using block of the context.
* Detached is even the state of entities retrieved from the database when tracking is disabled.
* When an entity is detached, it isn’t bound to the context, so its state isn’t tracked.
* It can be disposed of, modified, used in combination with other classes, or used in any other way you might need.
* Because there is no context tracking it, it has no meaning to Entity Framework.
* **Added State**
* When an entity is in the Added state, you have few options. In fact, you can only detach it from the context.
* Naturally, even if you modify some property, the state remains Added, because moving it to Modified, Unchanged, or Deleted makes no sense.
* It’s a new entity and has no correspondence with a row in the database.
* This is a fundamental prerequisite for being in one of those states (but this rule isn’t enforced by the context).
* **Modified State**
* When an entity is modified, that means it was in Unchanged state and then some property was changed.
* After an entity enters the Modified state, it can move to the Detached or Deleted state, but it can’t roll back to the Unchanged state even if you manually restore the original values.
* It can’t even be changed to Added, unless you detach and add the entity to the context, because a row with this ID already exists in the database, and you would get a runtime exception when persisting it.
* **Deleted State**
* An entity enters the Deleted state because it was Unchanged or Modified and then the DeleteObject method was used.
* This is the most restrictive state, because it’s pointless changing from this state to any other value but Detached.
  1. **Entity Framework Approach types:**
* Code First
* Database First
* Model First