**LINQ**

Before LINQ, we used various languages and technologies to access different data sources.

Now with LINQ we can use the same syntax to access the different data sources from within the language itself.

Since LINQ is a part of language it benefits from strong typing features like compile time checking for errors. Without LINQ, we used to pass SQL to the database at runtime and we get the error only at runtime, which is difficult to debug.

**Code First Approach**

Code first approach we have the code as bunch of POCO classes but we do not have Database.  Through the help of code, only it will generate the database. You have full control over on your code in code first approach.

Where code defines the database. Entity Framework handles creation. A complete detailed implementation of Entity Framework Code First Approach

**DB First Approach**

Database First approach in Entity Framework is used to make an entity model from an existing database and decrease the amount of code that must be written

Regular approach used where database is first created or already exists.

**Model First**

In Model First approach in Entity Framework, the model classes and their relationships are created first with the help of **ORM**. Then the physical database is generated from the model. Therefore, a diagram of entity and relations are converted into the code model

Where model is drawn, first that further generate database scripts

**Explain How You Can Load Related Entities In Ef (entity Framework)?**

* Eager Loading
* Lazy Loading
* Explicit Loading

**Dbcontext and dbset in Entity Framework**

DbContext is a class in Entity Framework API that forms a connection between a domain or entity class and the database. It is primarily responsible for communicating with the database.

The DbSet is another class that represents an entity set to read, create, update, and delete operations. For the entities that connect to the database tables and views, the context class from DbContext has to contain the DbSet type properties.

**Eager Loading & Lazy Loading**

For example, we may have a User table that contains basic user information like UserId, Username, Password, emailaddress etc. and another table UserDetails that contains contact details of the same user with attributes like Id, UserId (Foreign key from Users table), Contact Address, State, City etc. So here, the user table has a one-to-many relation with the UserDetails table.

Entity Framework provides the ability to load the data of a parent entity as well as its related child entity at the same time

In other words, when we load the data of the user table, we also get the related data of each User from the UserDetails table. This is known as Eager Loading.

On the other hand, if we do not want to load the related entity data at the same time as when the main entity is fetched, we use Lazy Loading

**How to disable the lazy loading**

context.ContextOptions.LazyLoadingEnabled = false;

**How to handle transactions in Entity Framework**

The “SaveChanges()” method in Entity Framework operates within a transaction and saves results of the work.

**using (var transaction = context.Database.BeginTransaction())**

**try**

**{**

**Category c = new Category();**

**c.CategoryName = "Mobile";**

**context.Categories.Add(c);**

**context.SaveChanges();**

**Product p = new Product();**

**p.CategoryId = c.CategoryId;**

**p.ProductName = "HTC";**

**p.Price = "15000";**

**context.Products.Add(p);**

**context.SaveChanges();**

**transaction.Commit();**

**}**

**catch (Exception ex)**

**{**

**transaction.Rollback();**

**}**