Sometimes, we will need to create a record in a Parent object and it's related record in a Child object (**Master-Detail relationship or Lookup relationship**) in a Salesforce Apex class or trigger. Let's say, we have a trigger in the Opportunity object which will create a Closed Won opportunity and Invoice together with the related invoice line items:

Invoice__c [Parent] **<-** Parent_Invoice__c (Master-Details Lookup custom field) **<-** Invoice_Line__c [Child]

We know that the Parent_Invoice__c custom field will only accept the Invoice Id. So, the easiest way to create and associate the Parent and Child record in this scenario is:
1. insert the Invoice__c record
2. get the new Invoice__c record Id returned from the insert
3. assign the new Invoice__c record Id to the Parent_Invoice__c custom field in a new Invoice_Line__c record
4. insert the Invoice_Line__c record

```
for(Opportunity opp : Trigger.new){
    Invoice__c newInvoice = new Invoice__c();
    ...
    ...
    //insert parent record here
    insert newInvoice;

    ...
    ...
    for(OpportunityLineItem oli : oppLineItems){
        Invoice_Line__c newInvLine = new Invoice_Line__c();

        // assign the Invoice record Id to Parent_Invoice__c custom field
        newInvLine.Parent_Invoice__c = newInvoice.Id;
        ...
        ...
        invLinesList.add(newInvLine);
    }

    //insert child record here
    insert invLinesList;
}
```

Yes, you can achieve the result that you want. But, you will encounter the Salesforce DML governor limit and performance issues if you have more than 200 opportunities in a batch or someone is doing a **BULK load into Salesforce.**
Don't worry, here is an elegant way of inserting the Parent and Child record without the need to use the Salesforce Id, instead using an **External Id field** in Parent object. Creating an External Id custom field in a Salesforce object will not only allow you to upsert data into Salesforce without the Salesforce Id, but it will also act as a Foreign Key for all related objects.

```
for(Opportunity opp : Trigger.new){
    Invoice__c newInvoice = new Invoice__c();
    newInvoice.External_Id__c = 'invoice_unique_id'; //e.g, INV0001
    ...
    ...
    newInvoicesList.add(newInvoice);

    ...
    ...
    for(OpportunityLineItem oli : oppLineItems){
        Invoice_Line__c newInvLine = new Invoice_Line__c();

        // create a new invoice object and assign the same unique id, e.g, INV0001
        Invoice__c parentInvoice = new Invoice__c();
        parentInvoice.External_Id__c = 'invoice_unique_id';

        // assign the invoice object to the Parent_Invoice__r
        newInvLine.Parent_Invoice__r = parentInvoice;
        ...
        ...
        invLinesList.add(newInvLine);
    }
}

insert newInvoicesList;
insert invLinesList;
```

Can you see the difference? We assign a unique value to the External_Id__c custom field of Invoice__c record and the same unique value is used in the related Invoice_Line__c record. We also established the relationship by assigning the parentInvoice object to the Invoice_Line__c.Parent_Invoice__r relationship field (instead of Invoice_Line__c.Parent_Invoice__c which only accepts Salesforce Id).
Please note that the Invoice_Line__c.Parent_Invoice__c and the Invoice_Line__c.Parent_Invoice__r are still the same custom field, just that different a suffix will accept a different value:
__c takes Salesforce Id
__r takes Salesforce Object