ORM TECHNIQUE

ORM ---> Object Relationship & Mapping.

Creating a Custom Object / Table:

Object Name: Customer

1. Table: Customer__C (Standard Fields: Id, Name, Owner, CreatedBy, LastModifiedBy)

2. Tab (User Interface): Customers.

3. Business Logic (Class): Customer__C

Step 1: ORM will transform each salesforce object (table) in the form of a class, where the table name and class name are identical.

Ex: Account Table will be represented as "Account Class".

```
Public Class Account

{

        // Write the Business Logic

}
```

Opportunity Object will be represented as "Opportunity Class".

```
Public Class Opportunity

{

        // Write the Business Logic

}
```

Position__C object will be represented as "Position__C Class".

```
Public Class Position__C

{

        // Write the Business Logic

}
```

Hiring_Manager__C object will be represented as "Hiring_Manager__C Class".

```
Public Class Hiring_Manager__C

{

        // Write the Business Logic

}
```

Step 2: Each field inside the object will be represented in the form of a variable in the associated Class.

Ex: Account Object Fields will be represented as below

```
}

                    Public Class Account

                    {

                              Public ID Id;

                              Public String name, rating, industry, phone, fax, type,

                                                  ownership, website, accountNumber,
billingCity, billingStreet, billingCountry, billingState, billingPostalCode,
customerPriority__C, active__C,;

                              Public Decimal annualrevenue;

                              Public DateTime createdby, lastModifiedBy;

                              It also Provides 10 Standard Methods:

                                        1. Save()

                                        2. QuickSave()

                                        3. Cancel()

                                        4. Delete()

                                        5. First()

                                        6. Next()

                                        7. Previous()

                                        8. Last()

                                        9. HasPrevious()

                                        10. HasNext()

                    }

          Account acc = new Account();

          acc.Name = 'Ram Kumar';

          acc.Rating = 'Hot';

          acc.Industry = 'Banking';

          acc.Annualrevenue = 1000000;

          Insert acc;

          if(acc.id != Null)

          {

                    System.debug('Account Record Inserted successfully.');

                    System.debug('Account Record id is....: '+ acc.id);

          }


          Hiring_Manager__C hr = new Hiring_Manager__C();
```

```
hr.Name = 'Ram';

hr.Location__C = 'Bangalore';

hr.ContactNumber__C = '9900990088';

hr.Designation__C = 'HR Manager';

Insert hr;


if(hr.id != Null)

        System.debug('HR Record Inserted Successfully. '+ hr.id);
```

Note: We can use these classes to interact with the Salesforce Objects programmatically, to perform the DML Operations. We have to follow the below steps to interact with the Salesforce Objects.

Step 1: Create the Object of the Associated Class.

Ex: Account acc = new Account();

Step 2: Assign the values for each field inside the object.

Ex:
```
acc.Name = 'Ram Kumar';

acc.Rating = 'Hot';

acc.Industry = 'Manufacturing';

acc.Annualrevenue = 1500000;

acc.active__C = 'Yes';

....

....
```

Step 3: Insert the Record inside the object, by using "Insert Statement".

Ex: Insert acc;

Step 4: Print the Result of the Insertion.

Ex:
```
if(acc.id != Null)

System.debug('Account Record Inserted Successfully. ID is..: '+ acc.id);
```

Use Case: Write an apex program, to Insert a Lead Record inside the object.

Step 1: Create the object of the Associated Class

```
Lead ldRecord = new Lead();
```

Step 2: Supply the values for the Required fields.

```
ldRecord.FirstName = 'Praveen';

ldrecord.LastName = 'Kumar';

ldRecord.Rating = 'Hot';
```

```
                    ldRecord.Industry = 'Finance';
                    ldRecord.AnnualRevenue = 2300000;
                    ldRecord.Company = 'Honeywell Technologies Inc.';
                    ldRecord.Status = 'Open - Not Contacted';
                    ldRecord.leadSource = 'Phone';
                    ldRecord.Phone = '9900444477';
                    ldRecord.Fax = '900333333';
                    ldRecord.Email = 'praveen@gmail.com';
                    ldRecord.Street = '#401, Sai Residency';
                    ldRecord.City = 'Hyderabad';
                    ldrecord.State = 'Telangana';
                    ldRecord.country = 'India';
                    ldRecord.title = 'Sales Manager';
```

Step 3: Insert the Record, by using "Insert" statement

```
            Insert ldRecord;
```

Step 4: Print the Result of the Record Insertion

```
            if(ldRecord.id != Null)
System. Debug ('Lead Record has been Inserted successfully with Id...: '+ ldRecord.id );
```

Use Case: Write an apex program, to Insert a New Hiring Manager Record inside the object.

Step 1: Create the object of the Associated Class

```
      Hiring_Manager__C hrRecord = new Hiring_Manager__C();
```

Step 2: Supply the values for the Required fields.

```
                    hrRecord.Name = 'Aman Kumar';
                    hrRecord.Location__C = 'Pune';
                    hrRecord.Designation__C = 'HR Executive';
                    hrRecord.Email_Address_del__c = 'aman@gmail.com';
                    hrRecord.Contact_Number_del_del__c = '9944778866';
                    hrRecord.Current_ctc__C = 1800000;
```

Step 3: Insert the Record, by using "Insert" statement

```
            Insert hrRecord;
```

Step 4: Print the Result of the Record Insertion

```
            if(hrRecord.id != Null)
```

```
                System. Debug('Hiring Manager Record ID is....: '+ hrRecord.id);
```

Use Case: Write an apex program, to Insert a Contact Record inside the Object.

Step 1: Create the object of the Associated Class.

```
                Contact con = new Contact();
```

Step 2: Supply the values for the Required fields

```
                        con.FirstName = 'Mahesh';
                        con.LastName = 'Kumar';
                        con.title = 'Technical Lead';
                        con.Phone = '9900447788';
                        con.MobilePhone = '9955446677';
                        con.email = 'mahesh.kumar@gmail.com';
                        con.fax = '9900446655';
                        con.mailingStreet ='#301, Rama Residency, Banjara Hills, Road#12';
                        con.mailingCity = 'Hyderabad';
                        con.mailingState = 'Telangana';
                        con.mailingCountry = 'India';
```

Step 3: Insert the Record, by using "Insert" statement

```
                Insert con;
```

Step 4: Print the Result of the Record Insertion.

```
                if (con.id != Null)
        System. Debug('Contact Record Inserted Successfully. Record Id is....: '+ con.id);
```

Creating Related Records

By using Apex Programming, we can create related records inside the objects. i.e. we can create parent record and related child records inside the child objects, even though the objects may be associated with either Lookup / Master-Detail / Hierarchical / Many-Many relationships.

While Creating Related Records, we have to follow the below steps.

Step 1: Write the Code to Insert Parent Record.

Ex:

```
        Account acc = new Account();
        acc.Name = 'Parent Account';
        acc.Rating = 'Hot';
        acc.Industry = 'Education';
        acc.Active__C = 'Yes';
        ....
        ....
        Insert acc;
```

Step 2: Write the Code to Insert the Related Child Record.

```
Ex:     if(acc.id != Null)
        {
                System.debug('Account Record Inserted Successfully. '+ acc.id );
                Contact con = new Contact();
                con.FirstName = 'Test';
                con.LastName = 'Contact';
                con.Phone = '8899004455';
                con.Email = 'test@gmail.com';
                ....
                ....
                // Map the Contact to the Related Parent Account.
                con. AccountId = acc.id;
                Insert con;
                If (con.id! = Null)
                System. Debug('Contact Record Inserted Successfully.');
                }
```

Use Case: Write an apex program, to Insert an Account Record, and a Related Opportunity Record.

```
// Step 1: Write the Code to Insert Parent Record. (i.e. Account)
        Account acc = new Account();


         acc.Name = 'Tata Motors Inc.';
         acc.Rating = 'Hot';
         acc.Industry = 'Manufacturing';
         acc.AnnualRevenue = 25000000;
         acc.Phone = '9900445577';
         acc.Fax = '9900004455';
         acc.Ownership = 'Private';
         acc.Type = 'Customer - Direct';
         acc.Website = 'www.tatamotors.com';
         acc.CustomerPriority__C = 'High';
```

```
            acc.Active__C = 'Yes';
            acc.BillingCity = 'Hyderabad';
            acc.BillingState = 'Telangana';
           Insert acc;
// Step 2: Write the Code to Insert the Related Child Record (i.e. Opportunity Record)
           if(acc.id != Null)
             {
             System.debug('Account Record Inserted Successfully. Account Id is...: '+ acc.id);

        Opportunity oppty = new Opportunity();

           oppty.name = 'Test Opportunity';
           oppty.StageName = 'Prospecting';
           oppty.Amount = 57000;
           oppty.closedate = System.today();

           // Make the Opportunity should be related to Account.
           oppty.AccountId = acc.id;
             Insert oppty;
           if(oppty.id != Null)
            System.debug('Opportunity Record Inserted Successfully. Id is...: '+ oppty.id);
   }
```

Creating Bulk Records: Using Apex Programming, we can insert bulk records inside the object. i.e. we can write the piece of code to insert the record, and execute the same code required number of times based on the need using iterative statements.

```
Ex:     for(Integer counter = 1; counter <= 10; counter++)
        {
                Account acc = new Account();
                acc.Name = 'Test Account';
                acc.Rating = 'Hot';
                ....
                ....
                Insert acc;
        }
```

Use Case: Write an apex program, to insert 50 Hiring manager records inside the object.

```
        for(Integer counter = 1; counter <= 50; counter++)
    {
                // Write the Code to Insert a Hiring Manager Record.
                Hiring_Manager__C hrRecord = new Hiring_Manager__C();
                hrRecord.Name = 'Sample HR - '+ counter;
                if(counter < 25)
                hrRecord.Location__C = 'Bangalore';
        else
                hrRecord.Location__C = 'Pune';
                hrRecord.Designation__C = 'Recruitment Specialist';
                hrRecord.Email_Address__c = 'sampletest'+counter+'@gmail.com';
                hrRecord.Contact_Number__c = '9900447766';
                hrRecord.Current_ctc__C = 1500000;

                Insert hrRecord;
                if(hrRecord.id != Null)
 System.debug('Hiring Manager Record Inserted Successfully. Record Id is..: '+ hrRecord.id);
    }
```