SARASWATHY COLLEGE OF ENGINEERING AND TECHNOLOGY

OLLAKUR,TINDIVANAM - 604305

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BACHELOR OF ENGINEERING

2024-2025

FIFTH SEMESTER

SALESFORCE DEVELOPER

TEAM MEMBERS :

M.VISHVAKEDHU :421822104053

S.LAKSHMIPRIYA :421822104302

S.PERUMAL :421822104303

SARASWATHY COLLEGE OF ENGINEERING AND TECHNOLOGY

OLLAKUR,TINDIVANAM - 604305

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BACHELOR OF ENGINEERING

2024-2025

certified that this is a bonafide record of work done by

Name :M.VISHVAKEDHU

NM ID :57485ABFF483B079DEB4E1A7A821C79A

Universite Reg.no : 421822104053

Semester : 5

Branch : CSE

Year : 2024-2025

Staff-in-Charge **Head of the Department**

Submitted for the_____

Practical Examination held on_____

Internal Examiner **External Examiner**

SALES AUTOMOBILE USING SALESFORCE CRM

PROJECT VIEW:

This project is focused on creating a comprehensive Salesforce Automobile Information CRM, designed to address the challenges of managing and tracking automotive sales, inventory, and customer relationships in the automobile industry. The goal is to deliver a solution that leverages Salesforce CRM, cloud technologies, and automation to improve the management of customer interactions, streamline inventory tracking, and optimize sales processes. Through this project, we aim to enhance operational efficiency, customer experience, and data accuracy, while supporting the long-term goals of the automobile dealership or sales organization.

OBJECTIVES:

Business Goals:

- Streamline the management of customer data, vehicle inventory, and sales processes.
- Enhance customer engagement through personalized service and targeted marketing.
- Improve operational efficiency by automating routine tasks and workflows.

Specific Outcomes:

- Develop an integrated system for tracking vehicle inventory, customer inquiries, sales leads, and follow-ups.
- Implement automated workflows for sales, service, and customer support processes.
- Enable detailed reporting and analytics for performance monitoring and decision-making.
- Ensure seamless integration with other systems like financial and service management tools.

Salesforce Key Features and Concepts Utilized:

This section highlights the main Salesforce functionalities applied in this CRM system:

- **Salesforce Sales Cloud:** Used for managing leads, opportunities, and sales pipelines.
- **Salesforce Service Cloud:** Provides customer service functionality, including case management and customer support.
- **Custom Objects and Fields:** Created to track vehicle information (make, model, year, VIN, etc.) and manage inventory.
- **Apex Triggers and Classes:** Used to automate processes such as updating inventory status, sending notifications, or creating follow-up tasks for sales representatives.
- **Reports and Dashboards:** For real-time insights into sales performance, customer activity, and inventory levels.

- **Lightning Web Components (LWC):** Custom user interfaces for improved user experience and interaction with the CRM.

Detailed Steps to Solution Design:

Data Models:

Define custom objects like Vehicle Inventory, Customer, and Sales Opportunity to store all relevant data.

Establish relationships between objects such as Customer to Sales Opportunity and Vehicle Inventory to Sales Opportunity.

Design custom fields in each object (e.g., Vehicle Color, Model Year, Price, Customer Preferences).

User Interface Design:

- Create custom Lightning pages for sales reps, service agents, and management to easily access and update information.
- Utilize Lightning App Builder to create a user-friendly dashboard that displays key metrics such as active sales leads, available inventory, and customer interactions.

Business Logic:

- Define automation rules for sales and service processes (e.g., automating lead assignment, setting up reminders for follow-up, and creating tasks based on specific triggers).
- Use Flow and Process Builder to streamline sales workflows, such as sending automated emails to customers after a purchase or sending alerts when inventory is running low.

AUTOMOBILE INFORMATION OBJECT:

What Is an Object?

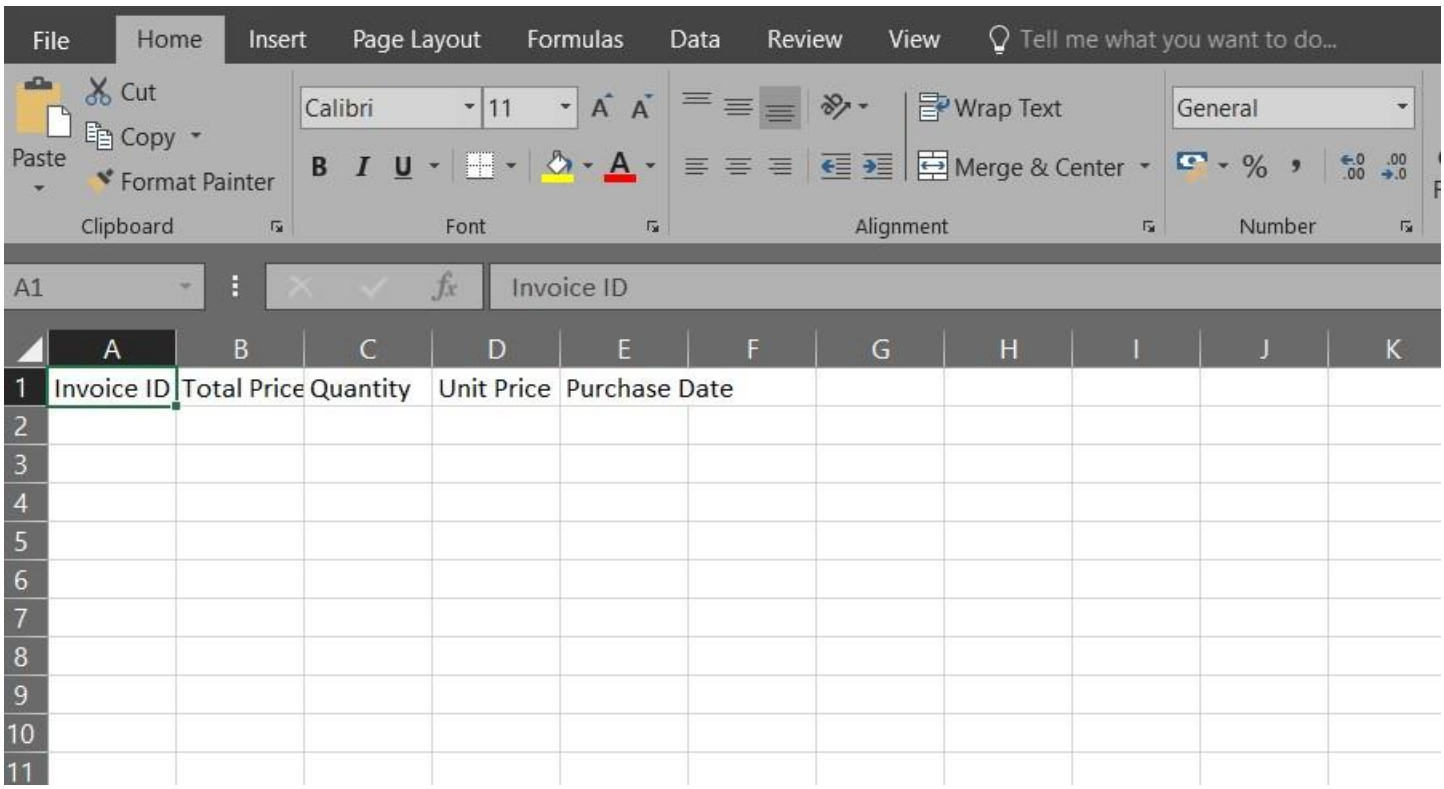
Salesforce objects are database tables that permit you to store data that is specific to an organization. What are the types of Salesforce objects

Salesforce objects are of two types:

1. **Standard Objects**: Standard objects are the kind of objects that are provided by salesforce.com such as users, contracts, reports, dashboards, etc.
2. **Custom Objects**: Custom objects are those objects that are created by users. They supply information that is unique and essential to their organization. They are the heart of any application and provide a structure for sharing data.

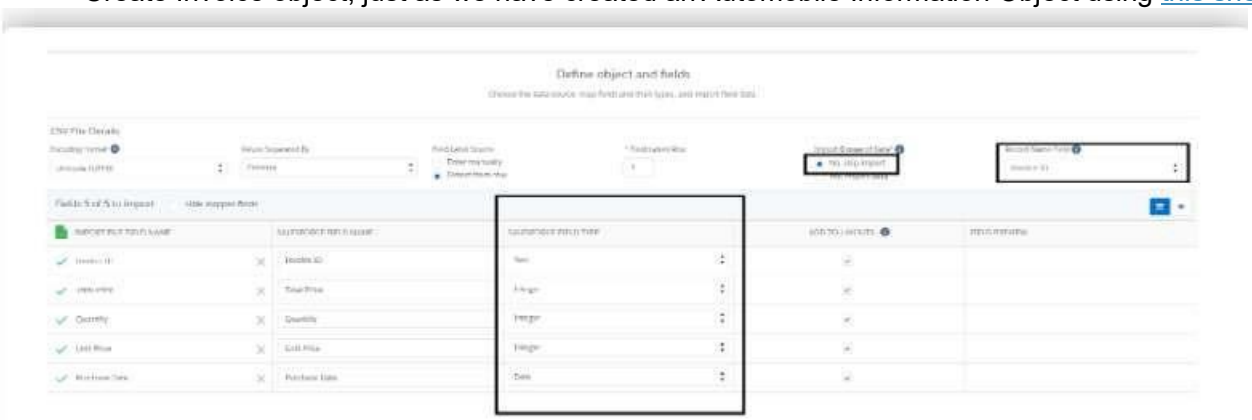
Create Automobile Information Object

1. Download and open this spreadsheet, save it as AutomobileInformation.csv.



Create Invoice Object

Create Invoice object, just as we have created an Automobile Information Object using this sheet



CREATING A CUSTOM TAB:

What is Tab: A tab is like a user interface that is used to build records for objects and to view the records in the objects.

Types of Tabs: **Custom Tabs**

Custom object tabs are the user interface for custom applications that you build in salesforce.com. They look and behave like standard salesforce.com tabs such as accounts, contacts, and opportunities.

Web Tabs

Web Tabs are custom tabs that display web content or applications embedded in the salesforce.com window. Web tabs make it easier for your users to quickly access content and applications they frequently use without leaving the salesforce.com application.

Visualforce Tabs

Visualforce Tabs are custom tabs that display a Visualforce page. Visualforce tabs look and behave like standard salesforce.com tabs such as accounts, contacts, and opportunities.

Lightning Component Tabs

Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app.

Lightning Page Tabs

Lightning Page Tabs let you add Lightning Pages to the mobile app navigation menu.

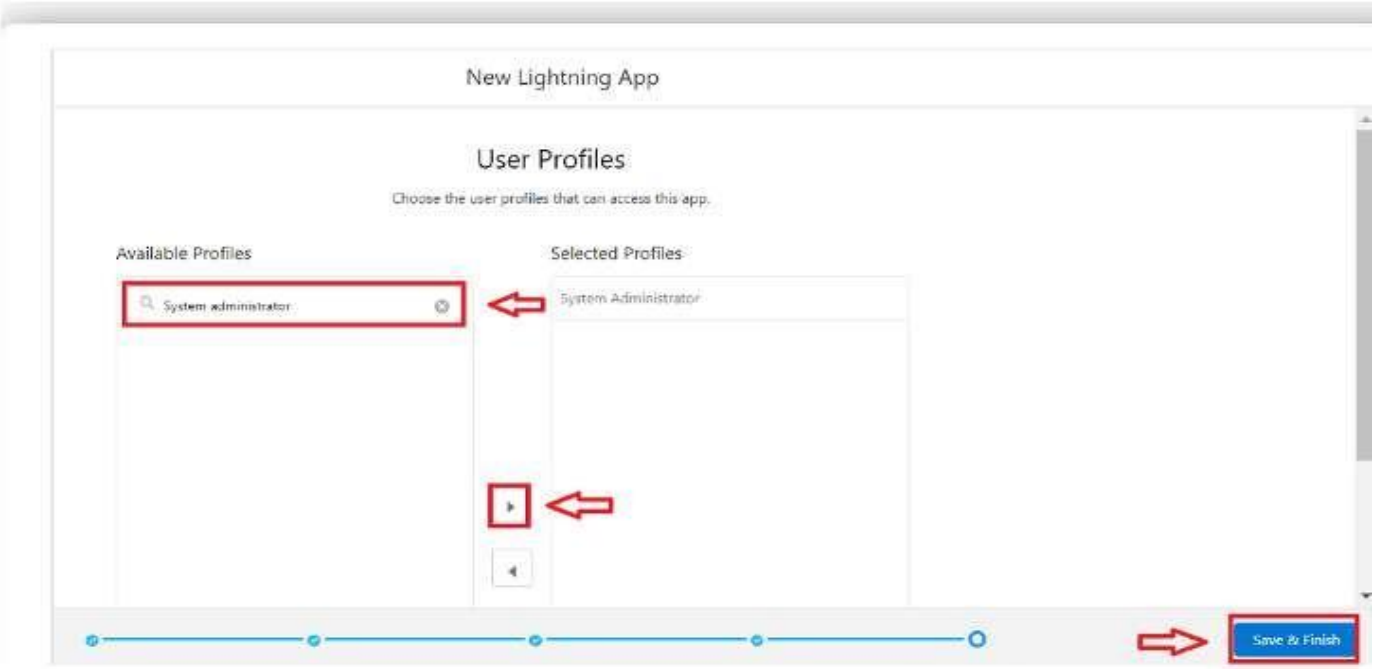
Lightning Page tabs don't work like other custom tabs. Once created, they don't show up on the All Tabs page when you click the Plus icon that appears to the right of your current tabs. Lightning Page tabs also don't show up in the Available Tabs list when you customize the tabs for your apps.

Creating a Custom Tab:

LIGHTNING APPS:

An app is a collection of items that work together to serve a particular function. In Lightning Experience, Lightning apps gives users access to sets of objects, tabs, and other items all in one convenient bund navigation bar. Lightning apps let you brand your apps with a custom color and logo. You can even include a utility bar and Lightning page tabs in your Lightning app. Members of your org can work more efficiently by easily switching between apps.

Create a Lightning App:



1.Fill the app name in app details and branding as follow

a. App Name :Sales Automobile Using Salesforce CRM
b. Developer Name : this will auto populated
c. Description : Give a meaningful description
d. Image : optional (if you want to give any image you can otherwise not mandatory)
e. Primary color hex value : keep this default

STEPS:

Search the items in the search bar(Account,Contact ,Opportunities,Automobile Information,Opportunity Automobile,Invoice, Reports, Dashboard) from the search bar and move it using the arrow button ? Next.
Note: select asset the custom object which we have created in the previous activity.

Search profiles (System administrator) in the search bar >> click on the arrow button >> save & finish.

Fields & Relationships

When we talk about Salesforce, Fields represent the data stored in the columns of a relational database. It can hold any valuable information that you require for a specific object. Hence, the overall searching, deletion, and editing of the records become simpler and quicker.

Types of Fields

1.Standard Fields

2.Custom Fields

As the name suggests, the Standard Fields are the predefined fields in Salesforce that perform a standard task. The main point is that you can't simply delete a Standard Field until it is a non-required standard field. Otherwise, users have the option to delete them at any point from the application freely. Moreover, we have some fields that you will find common in every Salesforce application. They are, >>Created By

>> Owner

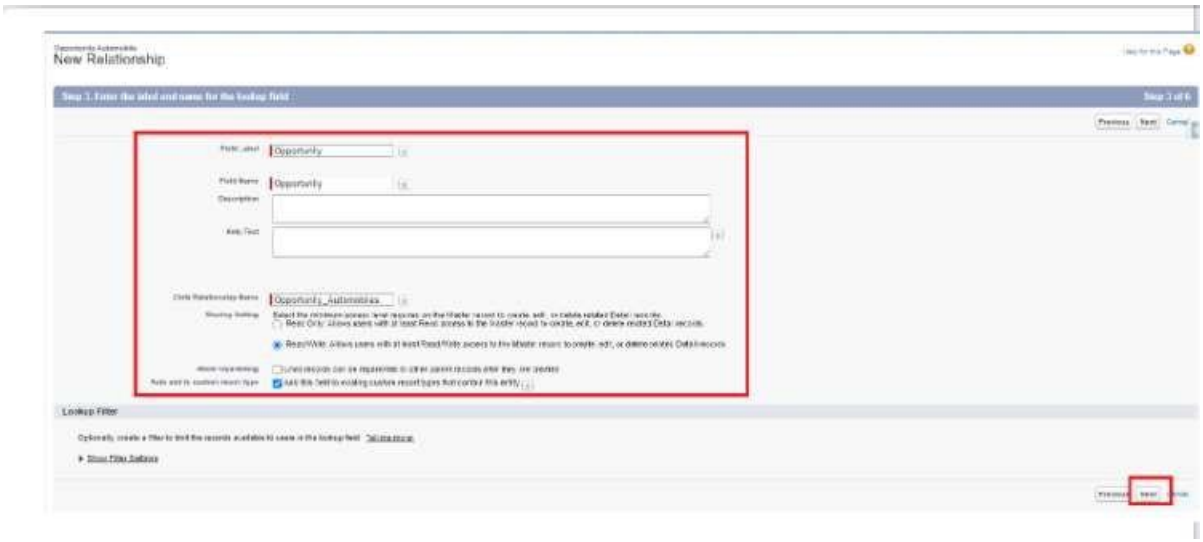
>> Last Modified

>> Field Made During object Creation

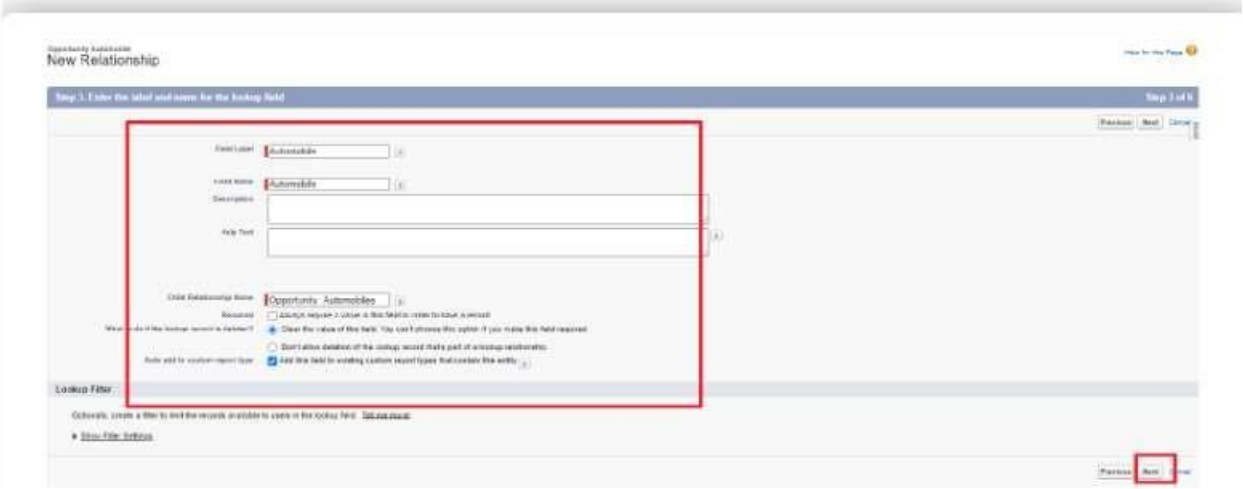
Custom Fields:

On the other side of the coin, Custom Fields are highly flexible, and users can change them according to requirements. Moreover, each organizer or company can use them if necessary. It means you need not always include them in the records, unlike Standard fields. Hence, the final decision depends on the user, and he can add/remove Custom Fields of any given form.

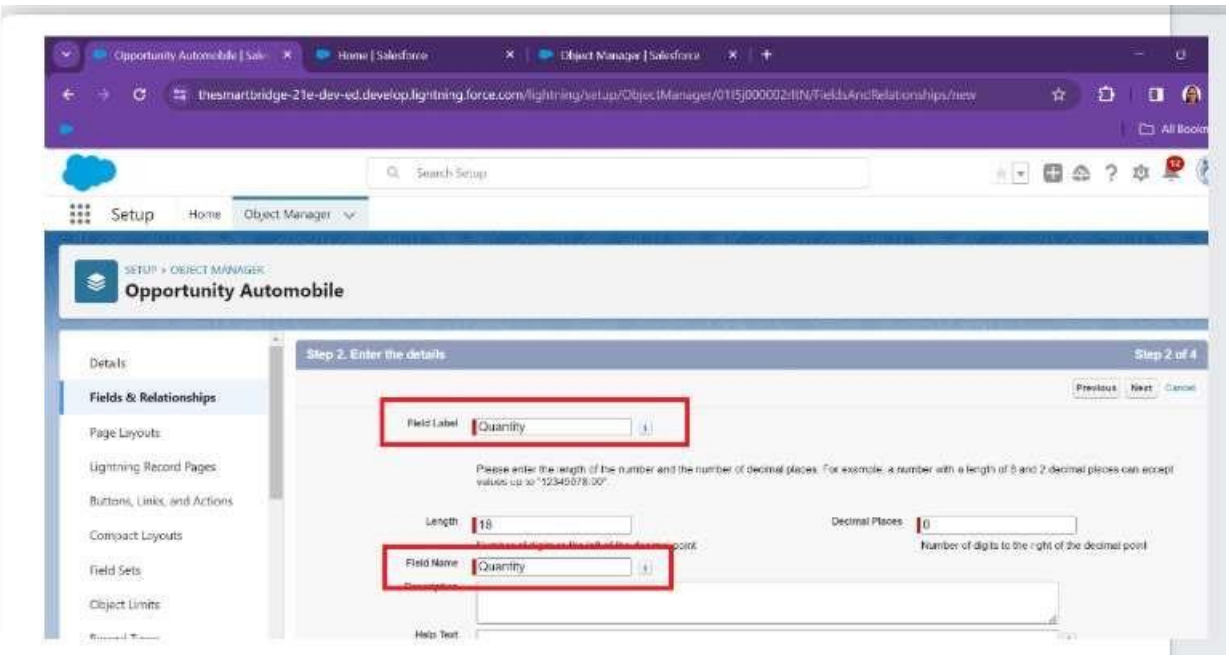
Creating Opportunity Master Detail Relationship Field in Opportunity AutoMobile Object:



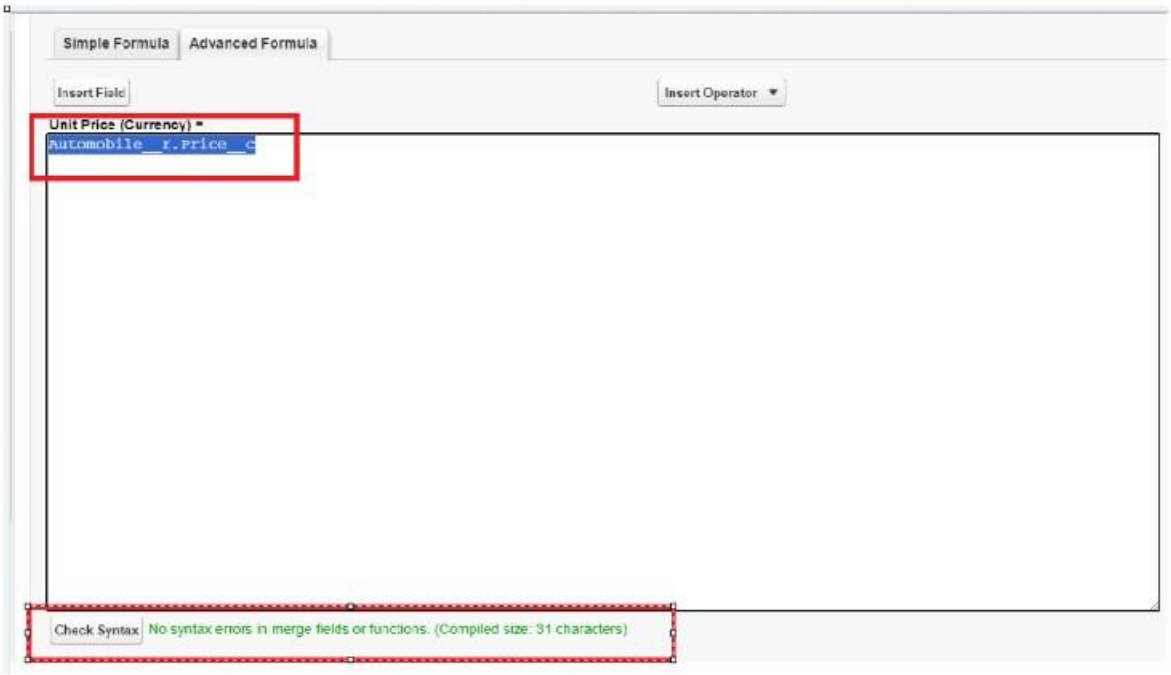
Creating the AutoMobile Information Lookup Field in Opportunity Automobile Object:



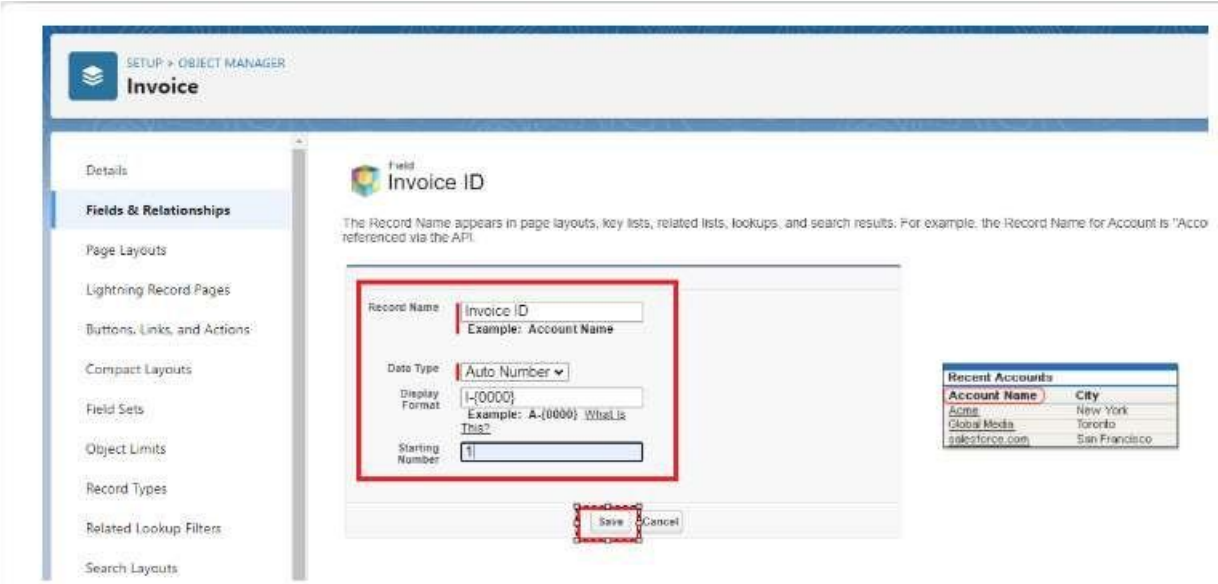
Creating Quantity Number Field in Opportunity Automobile Object



Creating Formula Field in Opportunity Automobile Object



Updating field in Invoice Object



Creating Remaining Fields in Objects

s.no	Object name	Fields	
1	Invoice	Field Name	Opportunity
		Data type	Master Detail relationship Object : Opportunity

Page Layouts:

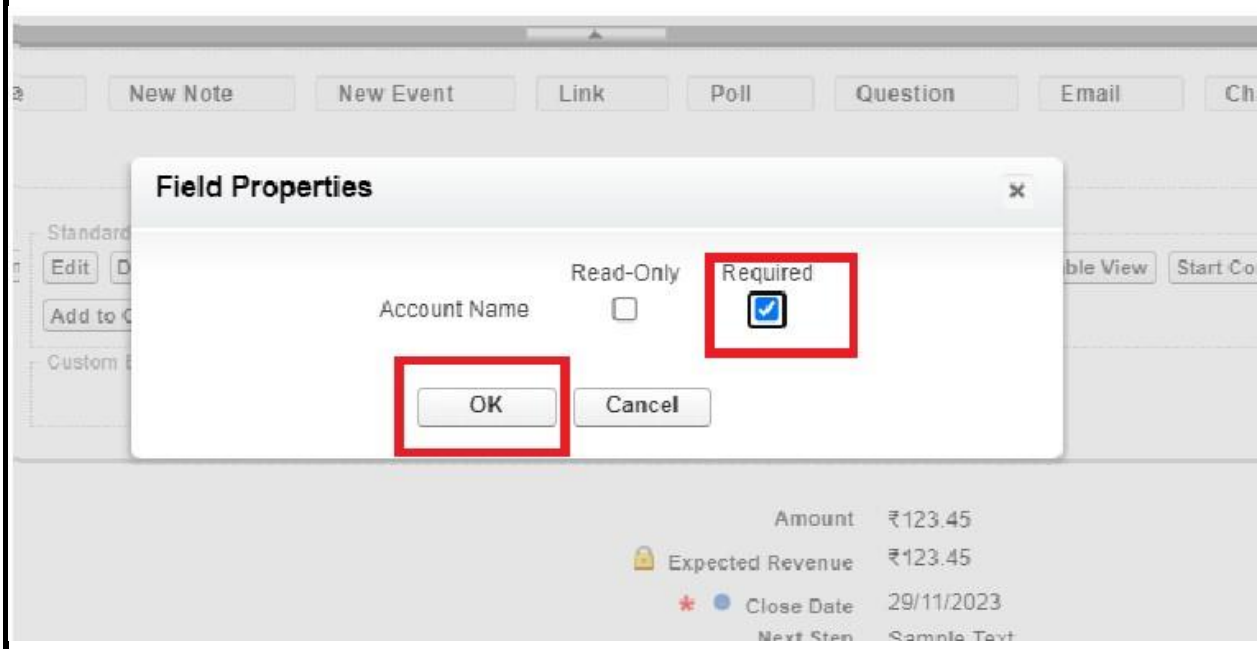
Page Layout in Salesforce allows us to customize the design and organize detail and edit pages of records in Salesforce. Page layouts can be used to control the appearance of fields, related lists, and custom links on standard and custom objects' detail and edit pages.

Edit the Page layout for Opportunity Object

Step 1: Go to Setup >> Click on Object Manager >> On the search bar, select Opportunity Layout. You can notice Page Layouts on the left panel

Step 2: Click on Page Layouts, Click on 'Opportunity Layouts'.

Step 3: In the Opportunity Detail Section, you can see various fields. Go on Account And Click on that Properties icon of Account name Field.

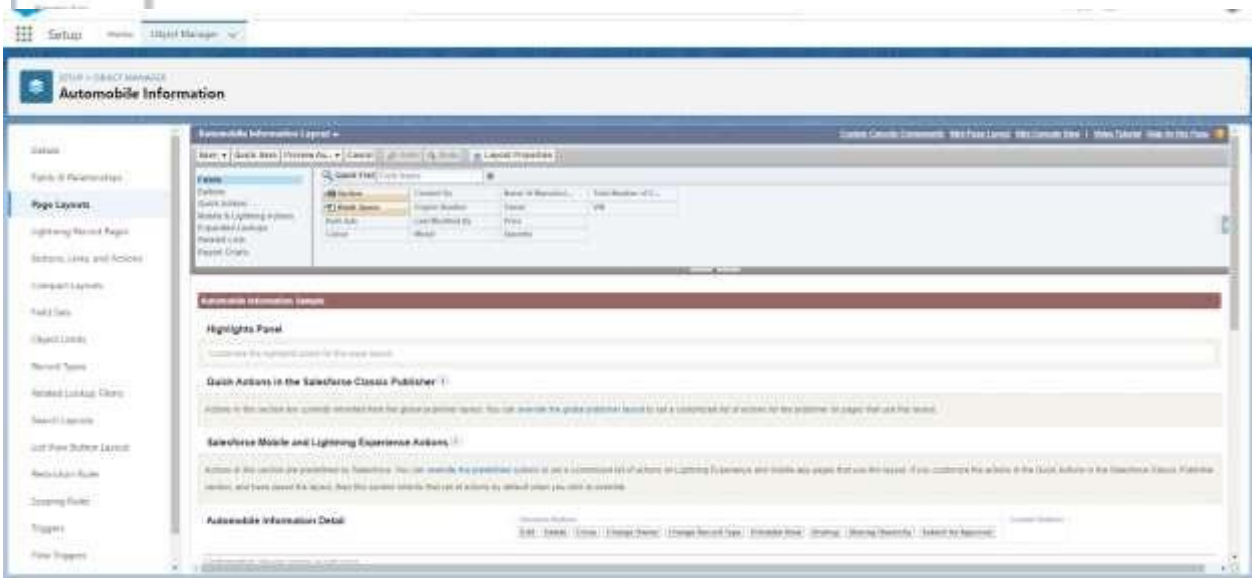


Step 4: check the Required box for Account name and click on Ok. Step 5: Click on Save.

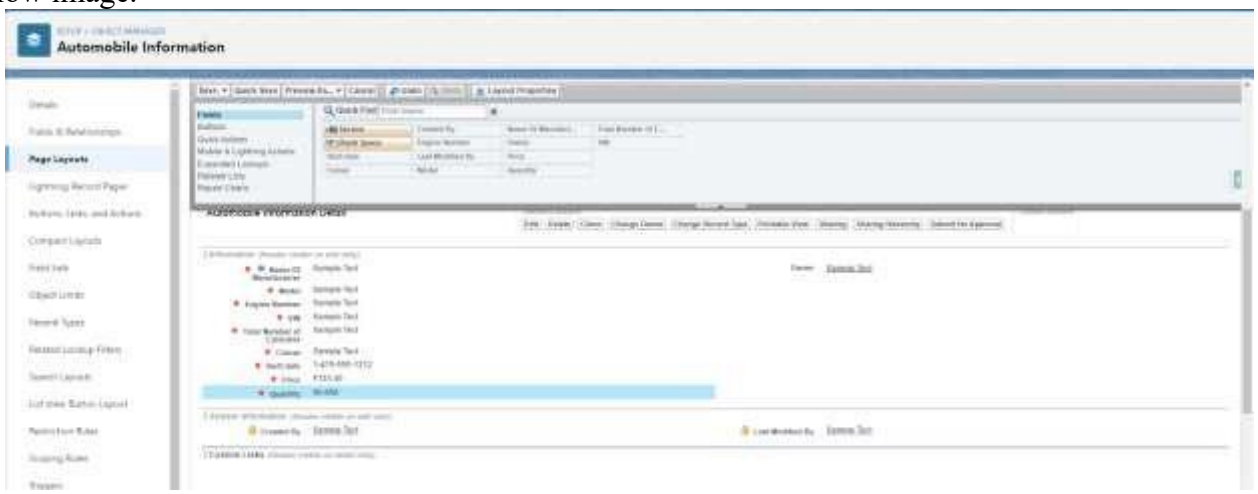
Edit the Page layout for Automobiles Information

Step 1: Go to Setup >> Click on Object Manager >> On the search bar, select Automobile Information. You can notice Page Layouts on the left panel

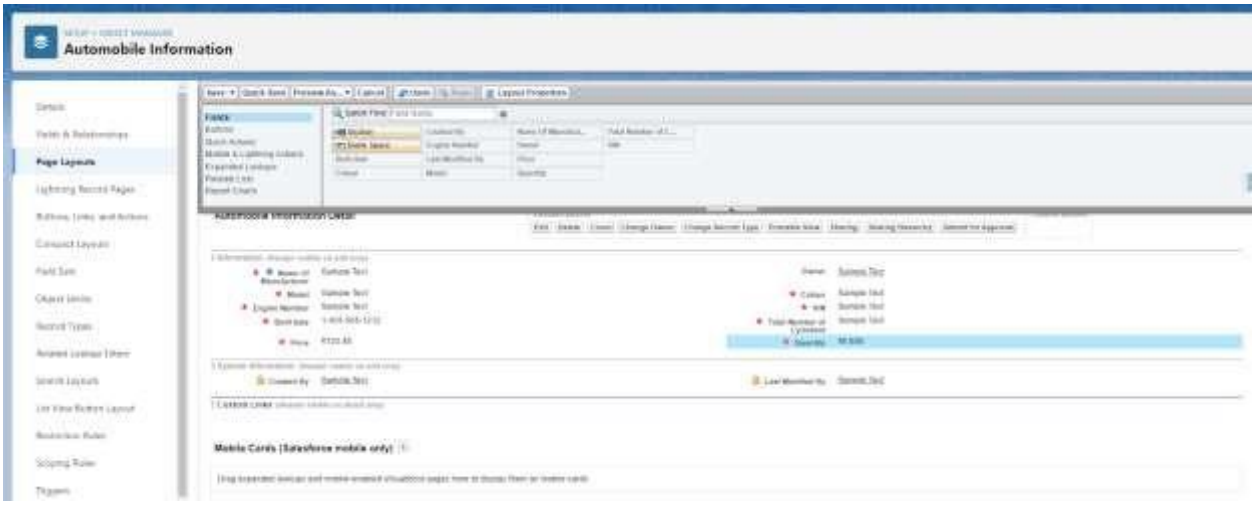
Step 2: Click on Page Layouts. Click on 'Automobile Information Layout'.



Step 3: Just Go for each one field of Automobile Information Object, Click on Gear Icon and mark as Required just as Done for Above Account Object. After required is done it will show the red color as given in below image.



Step 4 : Adjust the Fields as given below for A good looking view.



Step 5 : Click on Save.

Apex Trigger

Apex can be invoked by using triggers. Apex triggers enable you to perform custom actions before or after changes to Salesforce records, such as insertions, updates, or deletions. A trigger is Apex code that executes before or after the following types of operations:

insert update
delete merge
upsert
undelete

For example, you can have a trigger run before an object's records are inserted into the database, after records have been deleted, or even after a record is restored from the Recycle Bin.

You can define triggers for top-level standard objects that support triggers, such as a Contact or an Account, some standard child objects, such as a CaseComment, and custom objects. To define a trigger, from the object management settings for the object whose triggers you want to access, go to Triggers.

There are primarily two types of Apex Triggers:

Before Trigger: This type of trigger in Salesforce is used either to update or validate the values of a record before they can be saved into the database. So, basically, the before trigger validates the record first and then saves it. Some criteria or code can be set to check data before it gets ready to be inserted into the database.

After Trigger: This type of trigger in Salesforce is used to access the field values set by the system and affect any change in the record. In other words, the after trigger makes changes to the value from the data inserted in some other record.

Opportunity Automobile quantity

Code:

```
public class OpportunityHandlerClass {

public static void opportunityAutomobileQuantity(List<Opportunity> LstOpportunity, Map<Id,Opportunity>
 OldMapOpportunity){        set<Id>
opportunityIds = new set<Id>();
for(Opportunity opp : LstOpportunity){
if(opp.StageName =='Closed Won' ){
opportunityIds.add(opp.Id);
        }
    }
    Map<Id,Opportunity_Automobile__c> lstOpportunityAutomobile =new
  Map<Id,Opportunity_Automobile__c>([SELECT Id, Opportunity__c, Automobile__c, Quantity__c,
  Unit_Price__c, Total_Price__c FROM Opportunity_Automobile__c Where Opportunity__c IN:
  opportunityIds]);

    set<Id> AutoInformationIds = new set<Id>();
for(Opportunity_Automobile__c OppAuto: lstOpportunityAutomobile.values()){
if(OppAuto.Automobile__c != null){
            AutoInformationIds.add(OppAuto.Automobile__c);
        }
    }
    List<Automobile_Information__c> lstAutomobileInfomation = new List<Automobile_Information__c>();
    Map<Id,Automobile_Information__c> MapAutomobileInformation = New
 Map<Id,Automobile_Information__c>([SELECT Quantity__c, Price__c, Name, Id FROM
 Automobile_Information__c WHERE Id IN: AutoInformationIds]);
    For(Opportunity_Automobile__c AutoOpp : lstOpportunityAutomobile.Values()){            decimal
num = 0;            if(AutoOpp.Automobile__c ==
MapAutomobileInformation.get(AutoOpp.Automobile__c).Id &&
OldMapOpportunity.get(AutoOpp.Opportunity__c).stagename != 'Closed Won'){

            num = MapAutomobileInformation.get(AutoOpp.Automobile__c).Quantity__c- AutoOpp.Quantity__c;
MapAutomobileInformation.get(AutoOpp.Automobile__c).quantity__c = num;
            lstAutomobileInfomation.add(MapAutomobileInformation.get(AutoOpp.Automobile__c));
        }
    }
    If(!lstAutomobileInfomation.IsEmpty()){
update lstAutomobileInfomation;
```

```
    }


  }
```

Trigger Handler :

```
  trigger OpportunityTrigger on Opportunity (before update, After Update) {
if(trigger.isbefore && trigger.isUpdate){
      OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
  }
}
```

Opportunity-Automobile Error

```
  public class OpportunityAutomobileHandler {
  public static void quantityErrorOnAutomobileInformation(List<Opportunity_Automobile__c>
lstOpportunityAutomobile){
    set<Id> AutomobileIds = new Set<Id>();
    For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
if(oppAutomobile.Automobile__c != null){
          AutomobileIds.add(oppAutomobile.Automobile__c);
        }
    }
    Map<Id,Automobile_Information__c> lstAutomobileInformation = new
map<Id,Automobile_Information__c>([SELECT Id, CreatedById, Quantity__c, Price__c FROM
Automobile_Information__c WHERE Id IN: AutomobileIds]);
    For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
       If(OppAutomobile.Automobile__c == lstAutomobileInformation.get(OppAutomobile.Automobile__c).Id
  && lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c <
  OppAutomobile.Quantity__c){
          OppAutomobile.addError('the Number of Automobile u want are not Available !! the Automobile are
Available Count is ' +   .get(OppAutomobile.Automobile__c).Quantity__c );
        }
    }
  }
```

Trigger Handler :

```
trigger OpportunityAutoMobileTrigger on Opportunity_Automobile__c (before insert, before Update) {
  if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
     OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(trigger.new);
  }
}
```

Invoice Creation Trigger

```
  public class InvoiceCreation {
  public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity> lstOpportunity,
  Map<Id,Opportunity>OldMapOpportunity){
set<Id> oppIds = new Set<Id>();
For(Opportunity opp : lstOpportunity){
        if(Opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName !=
opp.StageName){
            oppIds.add(opp.Id);
        }
    }
    List<Opportunity_Automobile__c> lstOpportunityAutomobile = [SELECT Unit_Price__c, Total_Price__c,
  Automobile__c, Quantity__c,Opportunity__c, Id FROM Opportunity_Automobile__c WHERE Opportunity__c
  IN: oppIds];
    List<Invoice__c> lstInvoice = new List<Invoice__c>();
    For(Opportunity_Automobile__c oppAuto : lstOpportunityAutomobile){
        Invoice__c i = new Invoice__c();
        i.Quantity__c = oppAuto.Quantity__c;
        i.Unit_Price__c = oppAuto.Unit_Price__c;
        i.Total_Price__c = oppAuto.Total_Price__c;
        i.Purchase_Date__c = date.today();
        i.Opportunity__c = oppAuto.Opportunity__c;
lstInvoice.add(i);
    }
    if(!lstInvoice.isempty()){
insert lstInvoice;
    }
  }
```

Trigger Handler :

```
   trigger OpportunityTrigger on Opportunity (before update, After Update) {
  if(trigger.isbefore && trigger.isUpdate){
      OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
  }
  IF(trigger.isafter && trigger.isupdate){
```

```
    InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);
  }
```

Check contact role:

Trigger: public class

```
  ContactRoleCheck {
   public static void CheckcontactRoleonOpportunity(List<Opportunity> lstOpportunity,
  Map<Id,Opportunity>OldMapOpportunity){
     List<OpportunityContactRole> lstContactRole = [SELECT Id From OpportunityContactRole WHERE
OpportunityId IN: OldMapOpportunity.keyset()];        For(Opportunity opp : lstOpportunity){
         if(Opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName != opp.StageName){
If(lstContactRole.isempty()){                    opp.adderror('Please add contact Role on opportunity whenever
Opportunity is Going to Closed Won.');
            }
        }
     }
  } }
```

Trigger Handler :

```
  trigger OpportunityTrigger on Opportunity (before update, After Update) {
   if(trigger.isbefore && trigger.isUpdate){
       OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
ContactRoleCheck.CheckcontactRoleonOpportunity(trigger.new, trigger.oldMap);
  }
   IF(trigger.isafter && trigger.isupdate){
     InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);
  }
```

LWC Component:

Create Apex Class to Get Invoices:

```
  public class OpportunityInvoiceswithLWC {
  @AuraEnabled(cacheable=true)
  public static List<Invoice__c> getInvoices(string OpportunityId){
```

```
    return [SELECT Id, Quantity__c, Purchase_Date__c, Opportunity__c, Unit_Price__c, Total_Price__c, Name
FROM Invoice__c WHERE Opportunity__c =: OpportunityId];
}
```

}**Install Salesforce CLI:**

```
C:\Users\navee>sfdx
Salesforce CLI

VERSION
  sfdx-cli/7.182.1 win32-x64 node-v18.12.1

USAGE
  $ sfdx [COMMAND]

TOPICS
  alias    manage username aliases
  auth     authorize an org for use with the Salesforce CLI
  config   configure the Salesforce CLI
  force    tools for the Salesforce developer
  info     access cli info from the command line
  plugins  add/remove/create CLI plug-ins
  version                                          codekiat.com
```

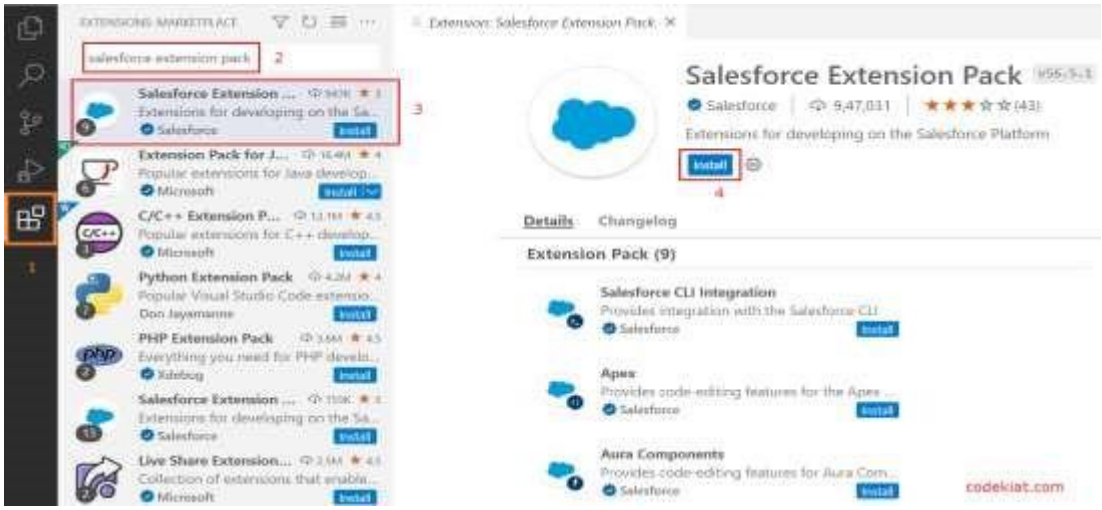
Install Microsoft VS Code:

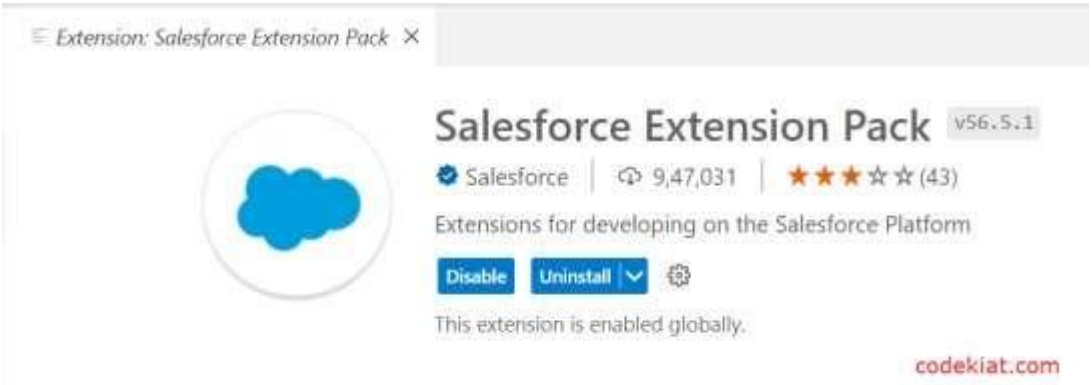
VS Code, or Visual Studio Code, is a free, open-source code editor developed by Microsoft. It is a lightweight, cross-platform code editor that provides features such as debugging, Git integration, and support for a wide range of programming languages.

Download the version of the software that is compatible with your operating system and install it.

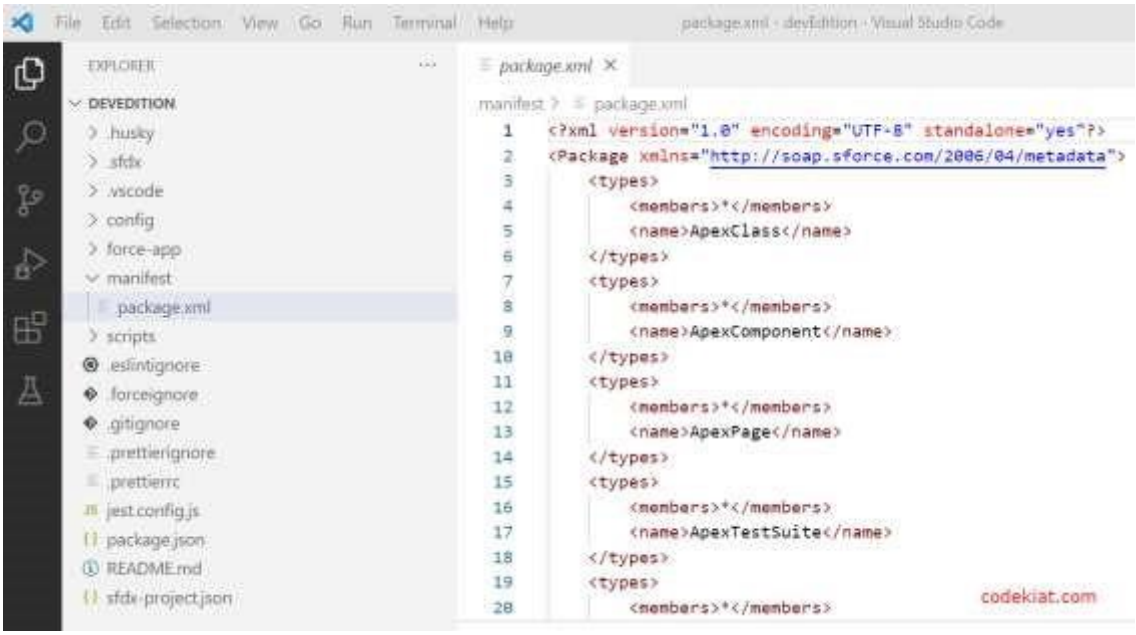
The following instructions are for Windows OS. Other operating systems may have slightly different steps.

Install the Salesforce Extension Pack:

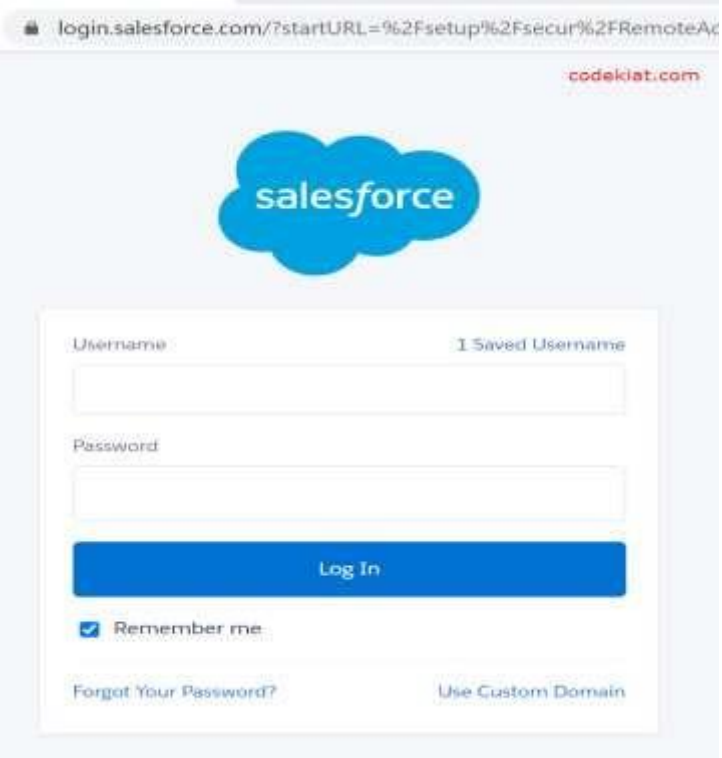




Create a project in VS Code:



Authorize an org:



Create Lightning Web Component:

XML File Code:

```
<?xml version="1.0" encoding="UTF-8"?>
  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>58.0</apiVersion>
<isExposed>true</isExposed>
 <targets>
    <target>lightning__RecordAction</target>
    <target>lightning__RecordPage</target>
 </targets>
  </LightningComponentBundle>
```

JS File Code :

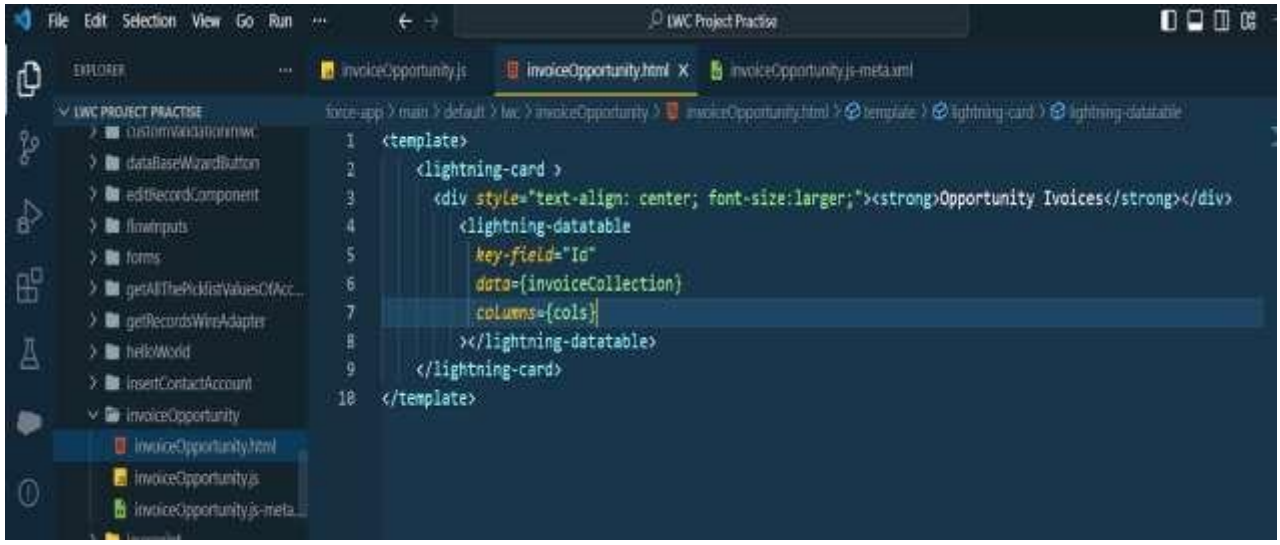
```
import { LightningElement, api, track,
wire } from 'lwc';
import getInvoices from '@salesforce/apex/OpportunityInvoiceswithLWC.getInvoices'; export
default class InvoiceOpportunity extends LightningElement {
@api recordId;
```

```
  @track invoiceCollection
cols = [
      {label:"ID" , fieldName:'Name'},
      {label:"Opportunity Id" , fieldName:'Opportunity__c'},
      {label:"Quantity" , fieldName:'Quantity__c'},

       {label:"Total Price" , fieldName:'Total_Price__c'},
        {label:"Purchase Date" , fieldName:'Purchase_Date__c'}
  ]

 @wire(getInvoices,{OpportunityId:'$recordId'})
invoicefunction({data,error}){
    console.log(this.recordId +'this is record Id');
if(data){
          console.log(data);
          this.invoiceCollection = data
        }if(error){
console.log('this is error')
         console.log('error');
        }
  }
```

HTML File :



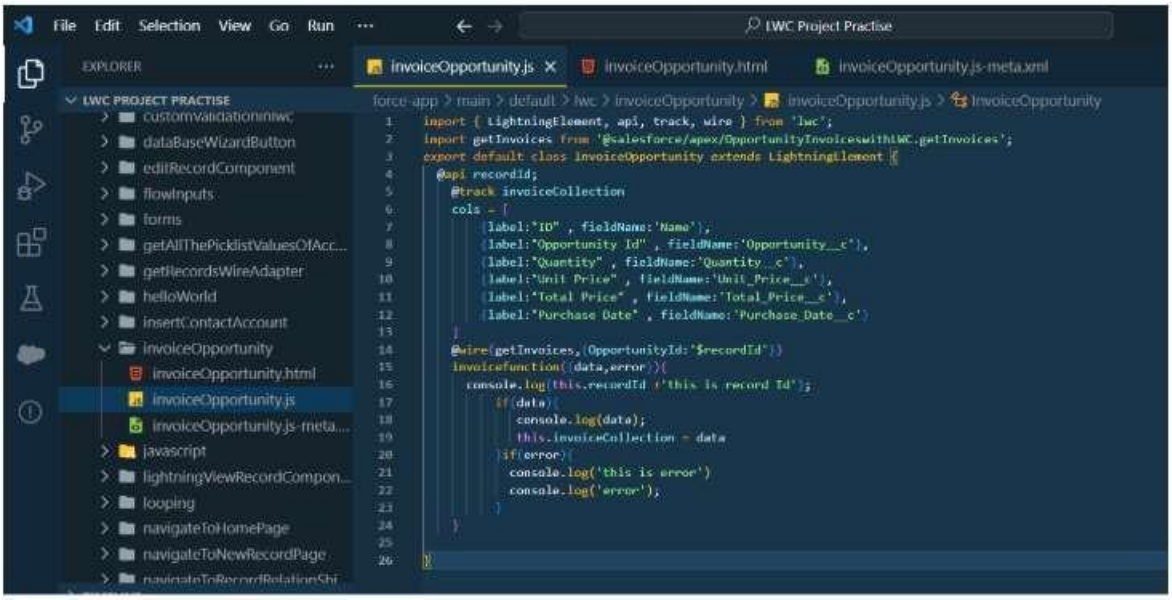

```
<template>
<lightning-card >
  <div style="text-align: center; font-size:larger;"><strong>Opportunity Ivoices</strong></div>
```

```
    <lightning-datatable key-field="Id" data={invoiceCollection} columns={cols}
    ></lightning-datatable>
  </lightning-card>
</template>
```

Create Lightning Web Component:

JS File Code :

```
import { LightningElement, api, track, wire } from 'lwc';
import getInvoices from '@salesforce/apex/OpportunityInvoiceswithLWC.getInvoices'; export
default class InvoiceOpportunity extends LightningElement {
@api recordId;
@track invoiceCollection
cols = [
    {label:"ID" , fieldName:'Name'},
    {label:"Opportunity Id" , fieldName:'Opportunity__c'},
    {label:"Quantity" , fieldName:'Quantity__c'},
    {label:"Unit Price" , fieldName:'Unit_Price__c'},
    {label:"Total Price" , fieldName:'Total_Price__c'},
    {label:"Purchase Date" , fieldName:'Purchase_Date__c'}
]
@wire(getInvoices,{OpportunityId:'$recordId'})
invoicefunction({data,error}){
  console.log(this.recordId +'this is record Id');
if(data){ console.log(data);
      this.invoiceCollection = data
    }if(error){
console.log('this is error')
      console.log('error');
    }
  }
}
```

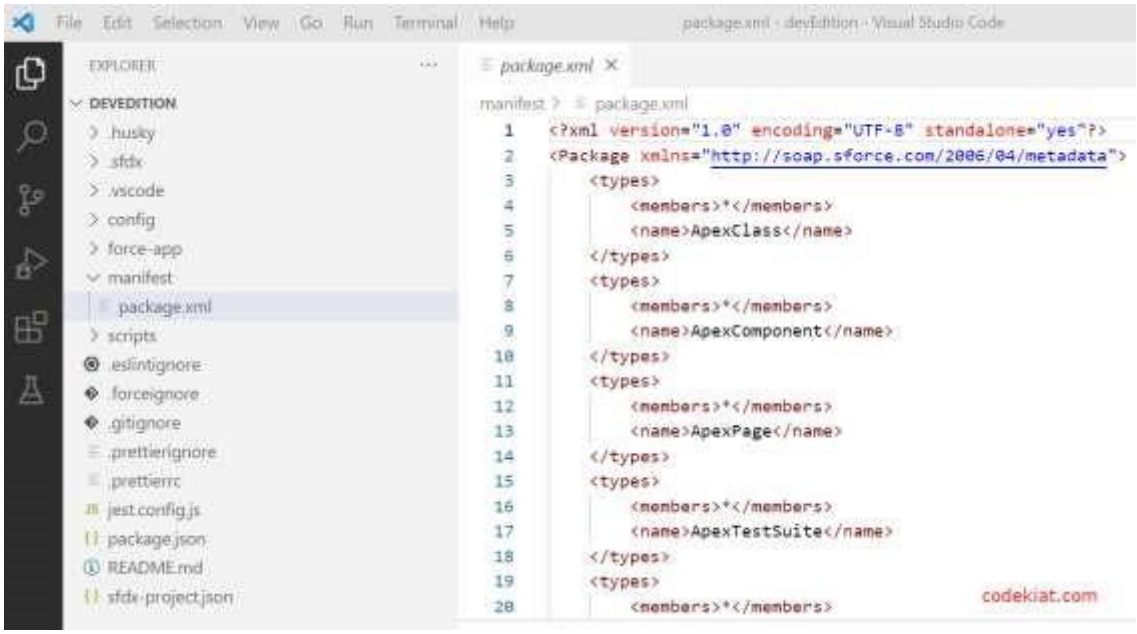


Create Button to Add on Opportunity

Select the InvoiceOpportunity component

Label :- Invoices

Name :- Invoices



Delete opportunity Schedule Class

Objective :

Through this schedulable class, we can see all the Closed Lost Opportunities.
We can delete all the Closed lost Opportunities by this Scheduled method on every monday as weekly.

1. Login to the respective account and navigate to the gear icon in the top right corner.
2. Click on the Developer console. Now you will see a new console window.
3. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
4. Name the class as “DeleteClosedLostOpportunities ”

CODE SNIPPET :

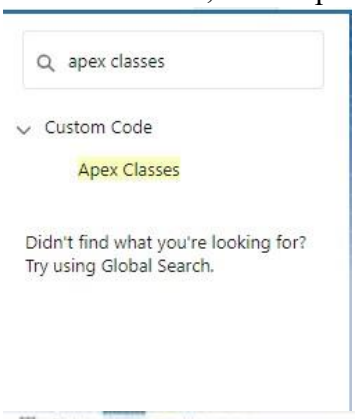
```
public class DeleteClosedLostOpportunities implements Schedulable{
    public static void execute(SchedulableContext sc){
        List<Opportunity> getLostOpportunities = [SELECT Id, Name From Opportunity Where StageName =: 'Closed Lost' LIMIT 50000];
        if(!getLostOpportunities.IsEmpty()){
            Delete getLostOpportunities;
        }
    }
}
```

```
public class DeleteClosedLostOpportunities implements Schedulable{
  public static void execute(SchedulableContext sc){
     List<Opportunity> getLostOpportunities = [SELECT Id, Name From Opportunity Where StageName =: '
Closed Lost' LIMIT 50000];
if(!getLostOpportunities.IsEmpty()){
        Delete getLostOpportunities;
     }
  }
}
```

Schedule the Apex class:

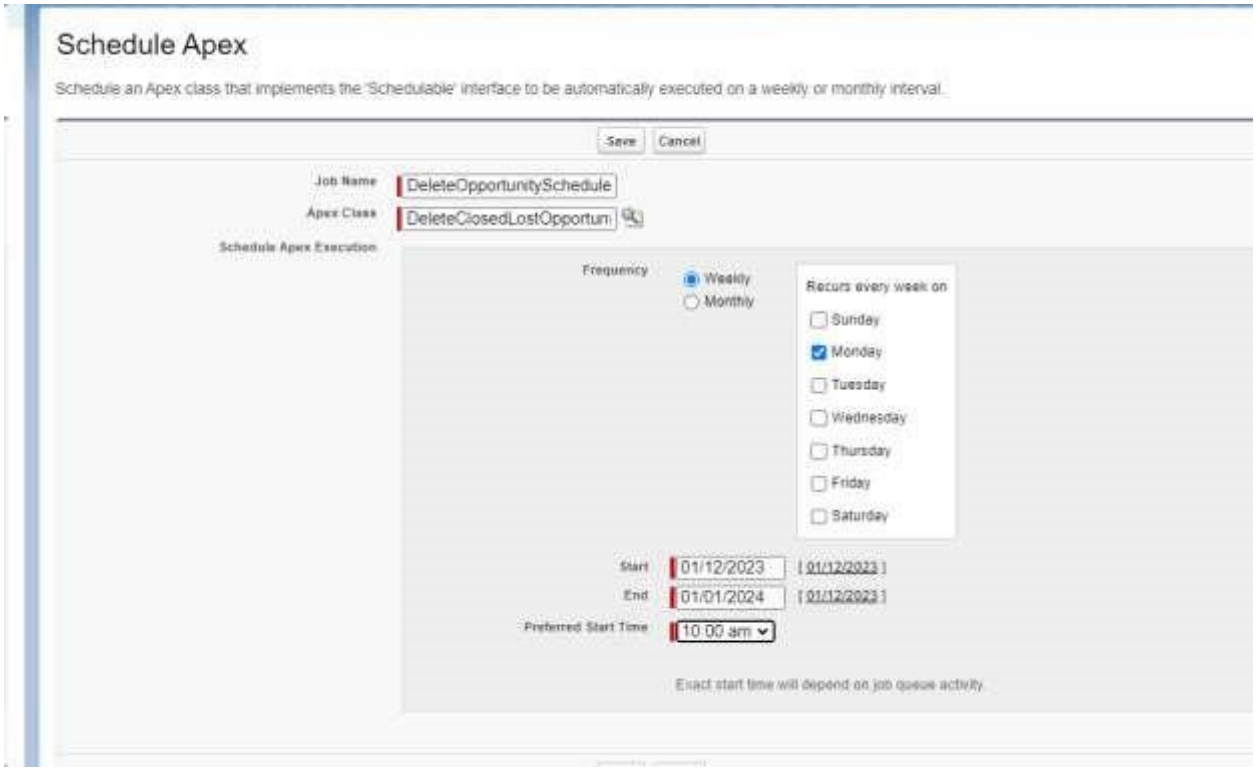
Go to the Home page in your salesforce account.

In the search bar, enter Apex and click on Apex Classes.

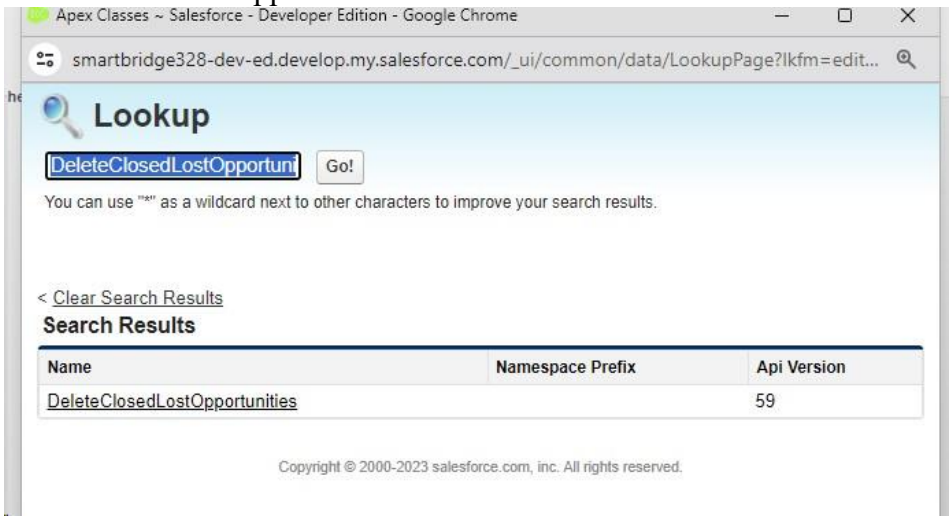


Click on Schedule Apex and enter the Job name. ○

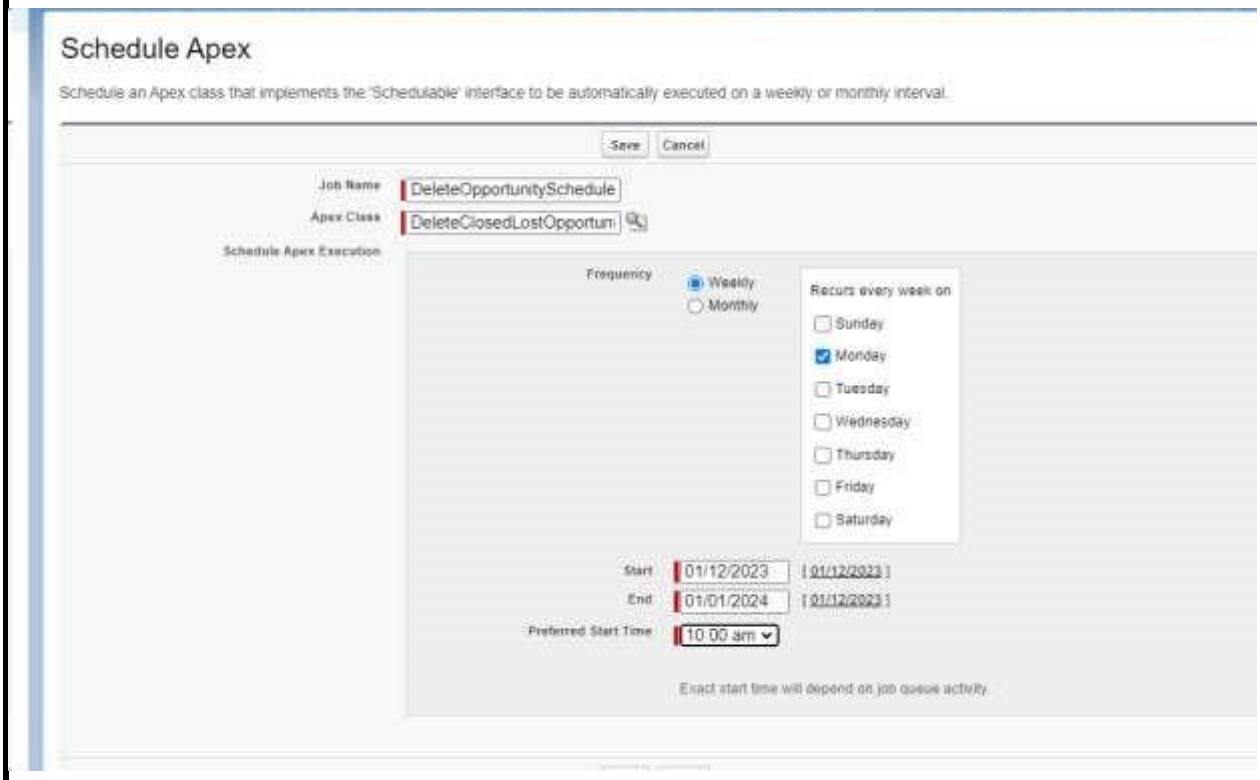
Job Name : DeleteOpportunitySchedule



Now click on the search icon present near the Apex class : Goto the Lookup icon beside ? click on it ? select DeleteClosedLostOpportunities.



In the Schedule Apex section , select weekly and select Monday mentioned and preferred time as 10:00 AM.



Click on Save. Now enter Apex in the search box and select Apex jobs.

You can see that the batch job is in queue and will run whenever the day mentioned comes.

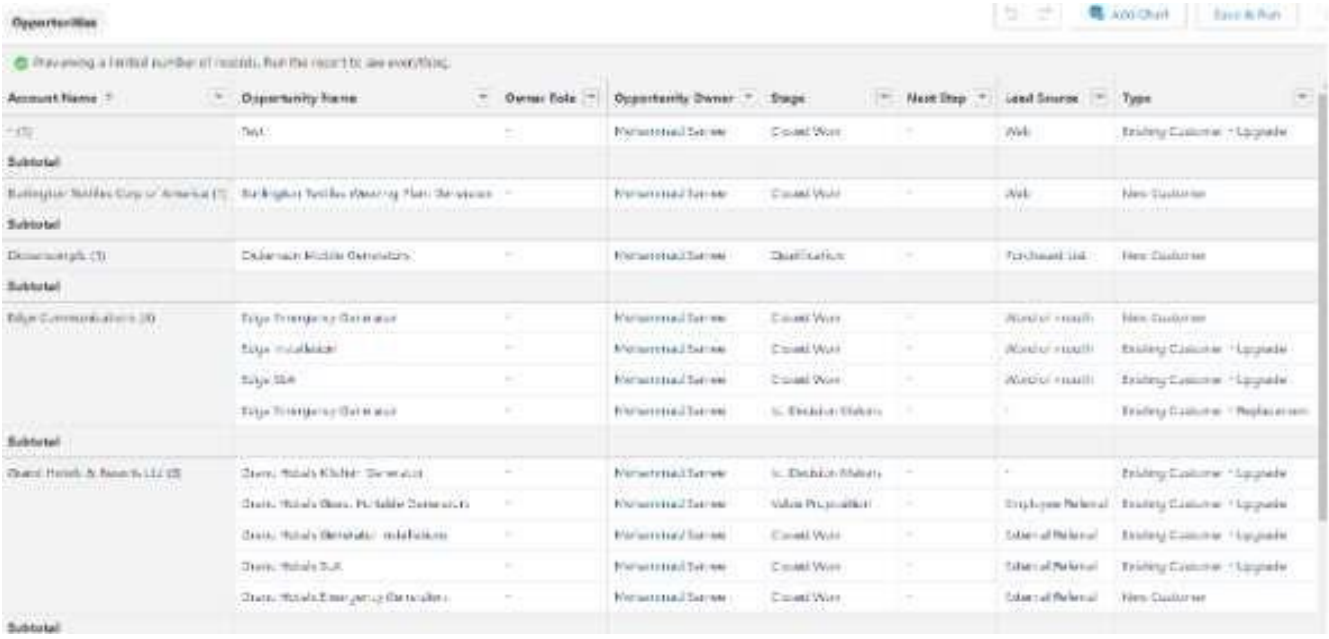
Reports:

Reports give you access to your Salesforce data. You can examine your Salesforce data in almost infinite combinations, display it in easy-to-understand formats, and share the resulting insights with others. Before building, reading, and sharing reports, review these reporting basics.

Types of Reports in Salesforce

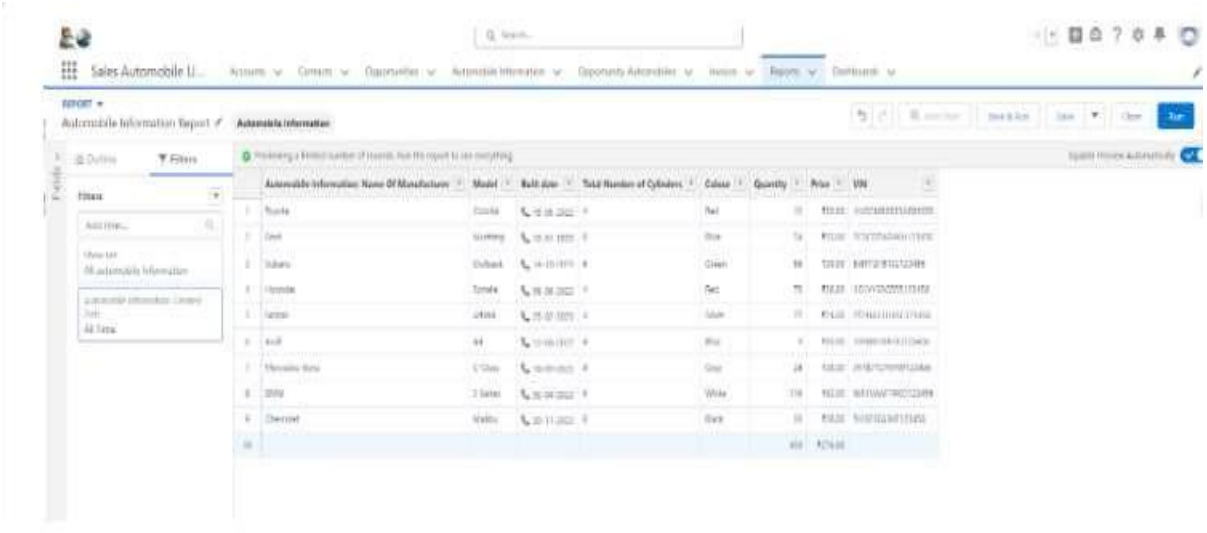
1. Tabular
2. Summary
3. Matrix
4. Joined Reports

Create Report on Opportunity:



Create Report on Automobile Information:

Filters:-



Dashboard:

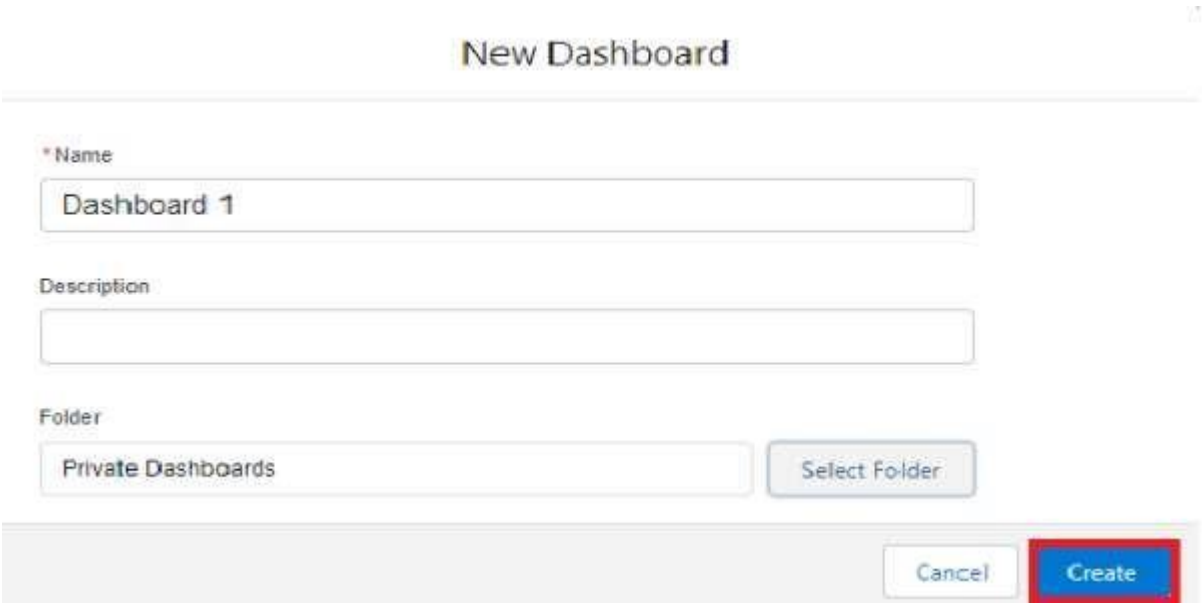
Dashboards help you visually understand changing business conditions so you can make decisions based on the real-time data you've gathered with reports. Use dashboards to help users identify trends, sort out quantities , and measure the impact of their activities. Before building, reading, and sharing dashboards, review these dashboard basics.

Sales Dashboard:

Create Dashboard

1. Go to the app ? click on the Dashboards tabs.

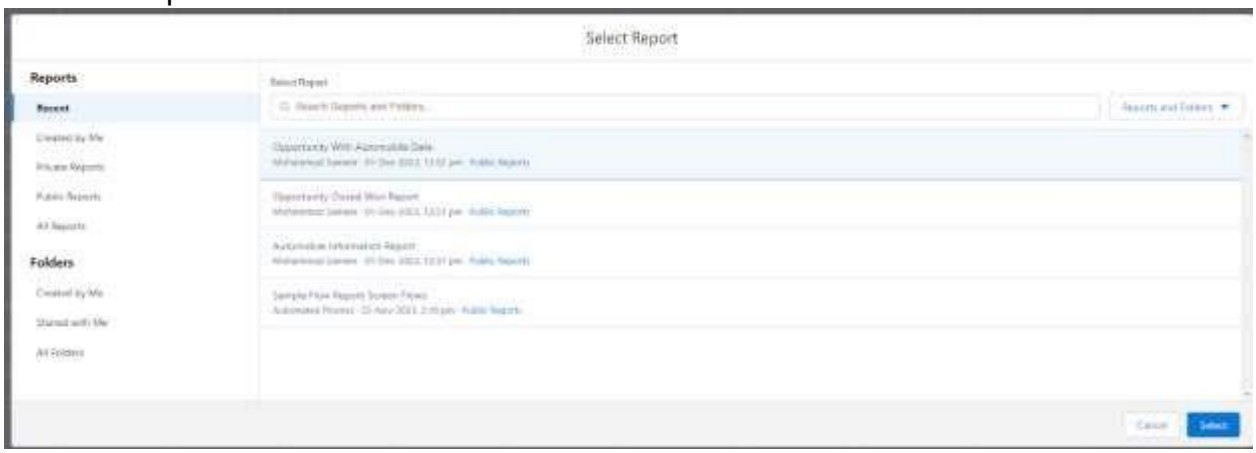
2. Give a Name and click on Create.



Name : Automobile Sales

3. Select add component

4. Select a Report and click on select.



5.Click Add then click on Save and then click on Done.

The Created Dashboard will look like this.

Conclusion:

Summary of Achievements:

The Salesforce Automobile Information CRM project has successfully integrated sales and service functionalities into a single platform, improving both internal operations and customer satisfaction. The CRM system now allows seamless tracking of inventory, sales opportunities, and customer interactions while automating key processes to boost efficiency. With enhanced reporting and analytics, the organization can make data-driven decisions to grow their business and improve the customer experience. The solution is scalable, customizable, and alignswith the long-term goals of the automobile dealership, setting a foundation for future growth and digital transformation.