

**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?**

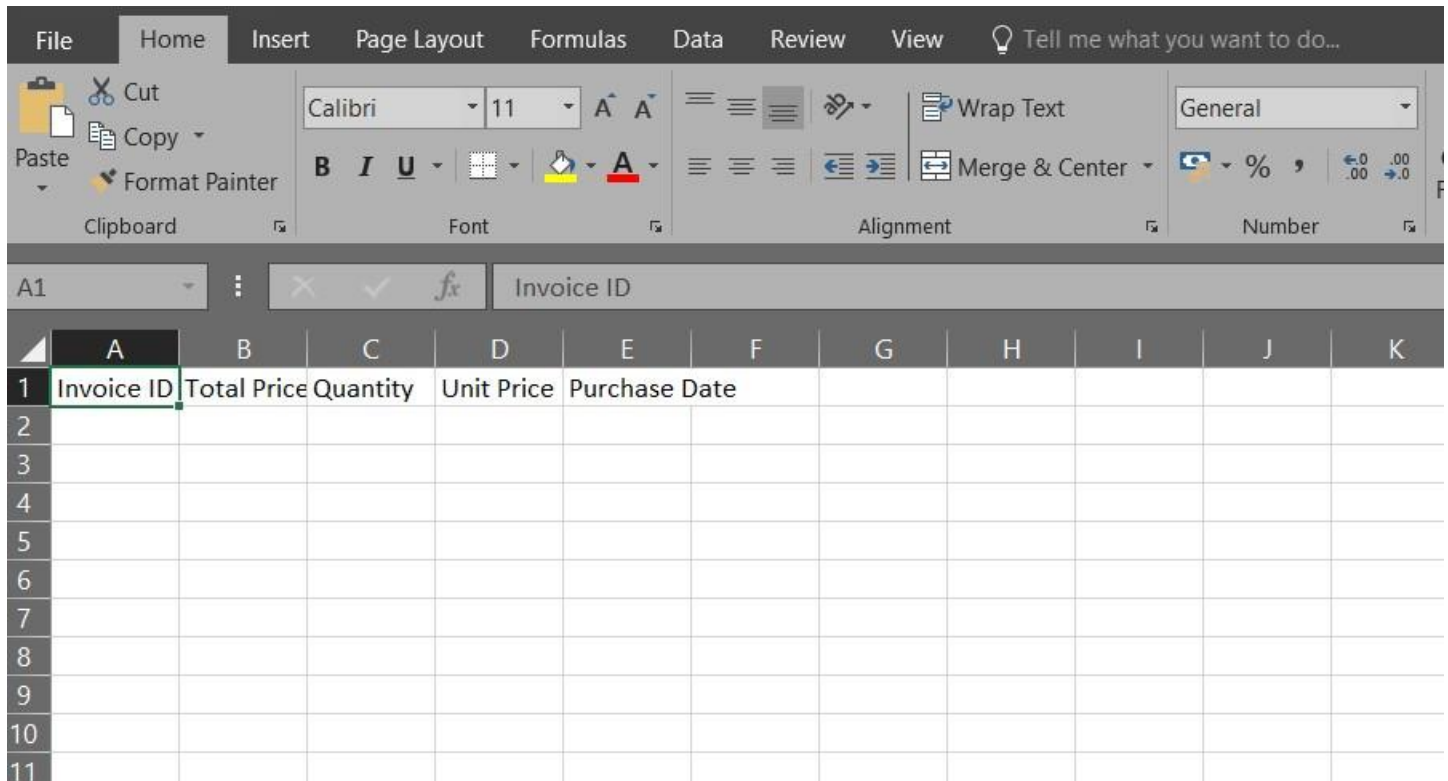
Salesforce objects are database tables that permit you to store data that is specific to an organization. What are the types of Salesforce objects

### **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

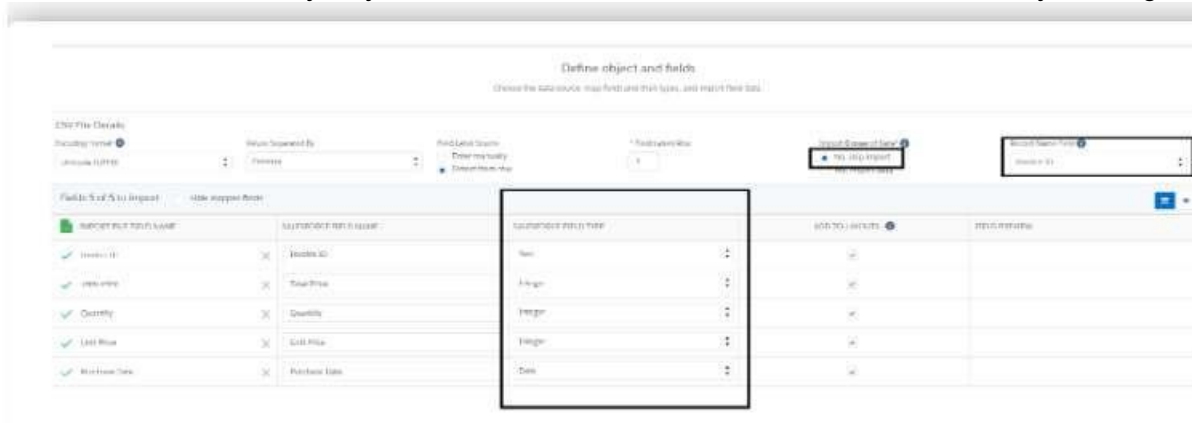
1. Download and open [this spreadsheet](#), save it as AutomobileInformation.csv.



	A	B	C	D	E	F	G	H	I	J	K
1	Invoice ID	Total Price	Quantity	Unit Price	Purchase Date						
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											

## Create Invoice Object

Create Invoice object, just as we have created an Automobile Information Object using [this sheet](#)



Define object and fields

Choose the table source, map fields and their types, and import their data.

Enter the Details

Object Name: **Invoice** (highlighted with a red box)

Table Source: **Invoice**

Field Selection: **Invoice ID**, **Total Price**, **Quantity**, **Unit Price**, **Purchase Date**

Field Types: **Text**, **Number**, **Integer**, **Integer**, **Date**

## 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.

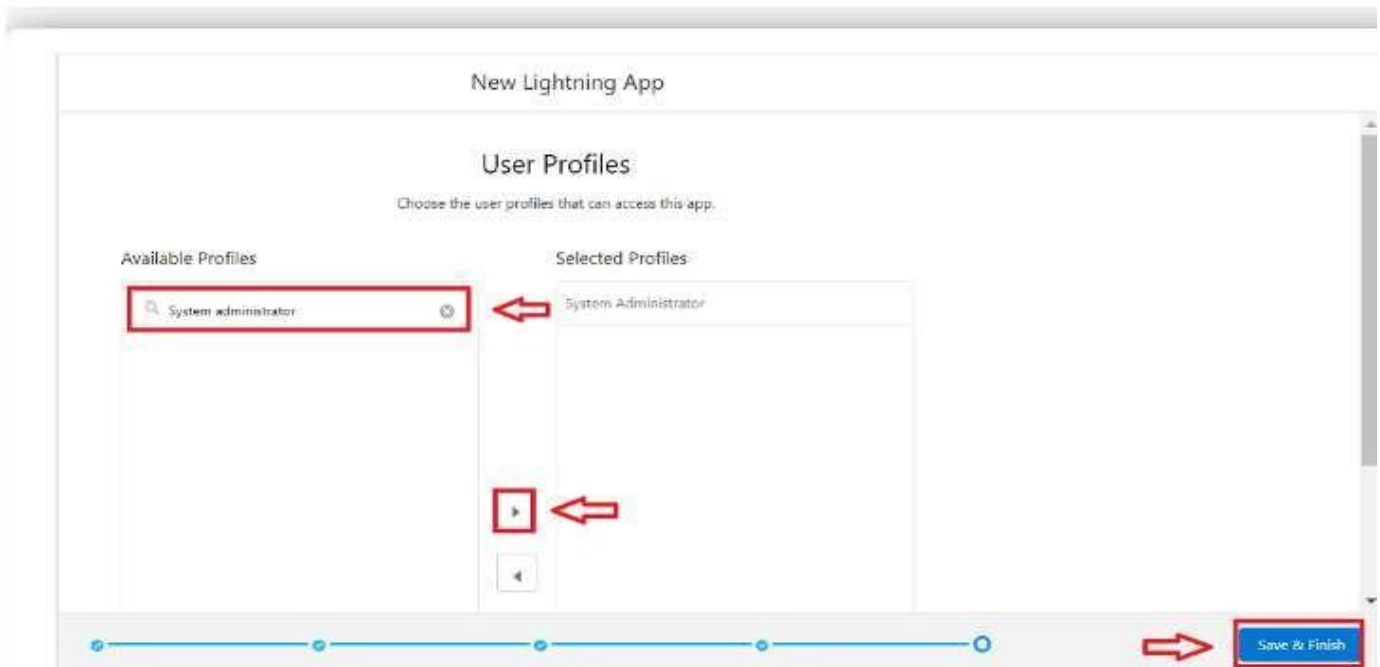
## **Creating a Custom Tab:**



## **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.

Step 3. Enter the info and name for this lookup field.

Field Label:

Field Name:

Description:

Field Type:

Create Relationship Name:

Sharing Settings: ☐ Read Only

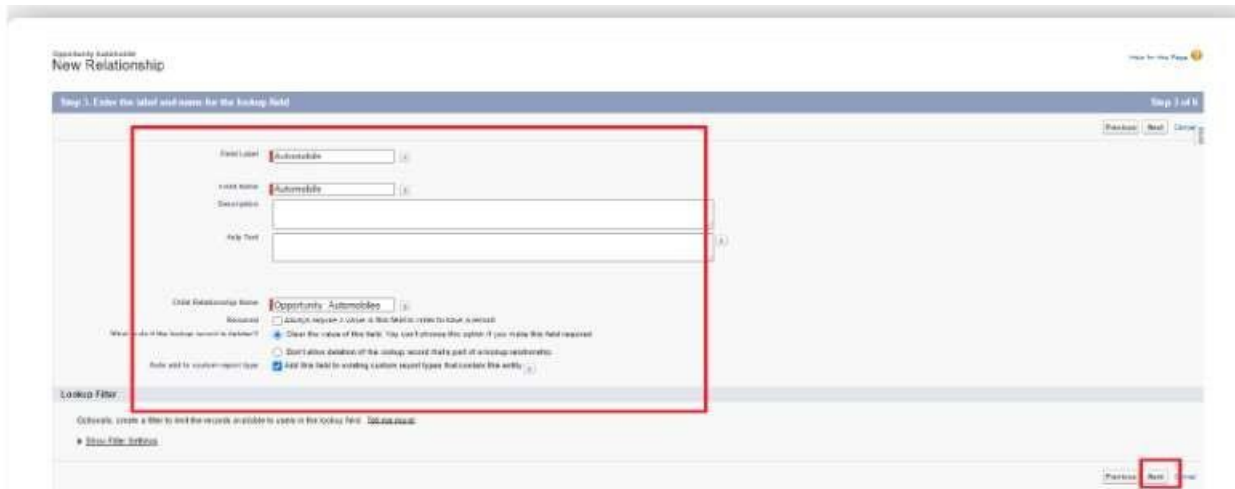
Allow Relationship: ☒ Allow Relationship

Lookup Filter: [Create Filter](#)



## Creating the AutoMobile Information Lookup Field in Opportunity

### Automobile Object:



Opportunity Automobile  
New Relationship

Step 1. Enter the label and name for the lookup field

Field Label: Automobile

Field Name: Automobile

Description:

Help Text:

Create Relationship Name: Opportunity Automobile

Relationship:

When used in the lookup search is required?

Clear the value of the field. You can't remove the value if you make this field required.

Don't allow deletion of the lookup record that's part of a lookup relationship.

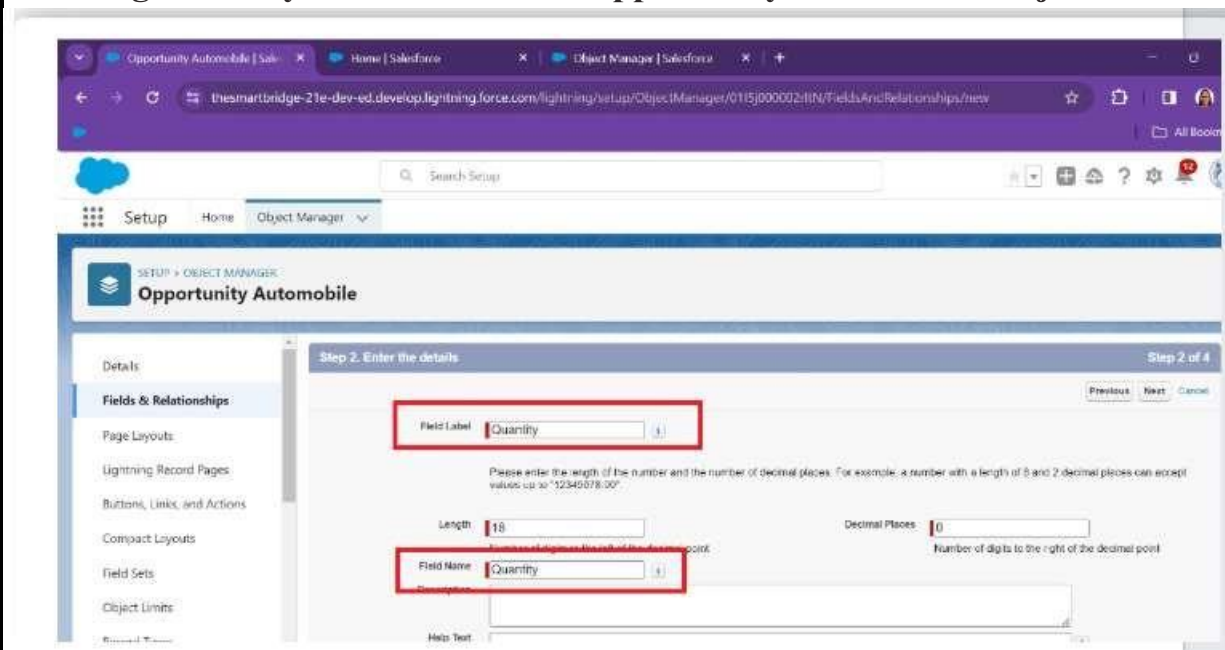
Add the field to existing custom report types that contain the entity.

Lookup Filter:

Click here to create a filter to limit the records available to users in the lookup field.

Previous Next Cancel

## Creating Quantity Number Field in Opportunity Automobile Object



Opportunity Automobile | Sales | Home | Salesforce | Object Manager | Salesforce

thesmartbridge-21e-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/0115j000002t1N/FieldsAndRelationships/new

Setup Home Object Manager

SETUP > OBJECT MANAGER

Opportunity Automobile

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Related Items

Step 2. Enter the details

Field Label: Quantity

Please enter the length of the number and the number of decimal places. For example, a number with a length of 8 and 2 decimal places can accept values up to "12345678.00".

Length: 18

Decimal Places: 0

Field Name: Quantity

Help Text:

Previous Next Cancel

## Creating Formula Field in Opportunity Automobile Object



Simple Formula Advanced Formula

Insert Field Insert Operator ▼

Unit Price (Currency) =  
Automobile\_\_r.Price\_\_c

Check Syntax No syntax errors in merge fields or functions. (Compiled size: 31 characters)

## Updating field in Invoice Object



SETUP > OBJECT MANAGER

**Invoice**

Details

**Fields & Relationships**

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

Field

**Invoice ID**

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name".

Record Name:  Example: Account Name

Date Type:

Display Format:  Example: A-0000 What is this?

Starting Number:

Recent Accounts	
Account Name	City
Acme	New York
Global Media	Toronto
salesforce.com	San Francisco

## Creating Remaining Fields in Objects

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

## Page Layouts:

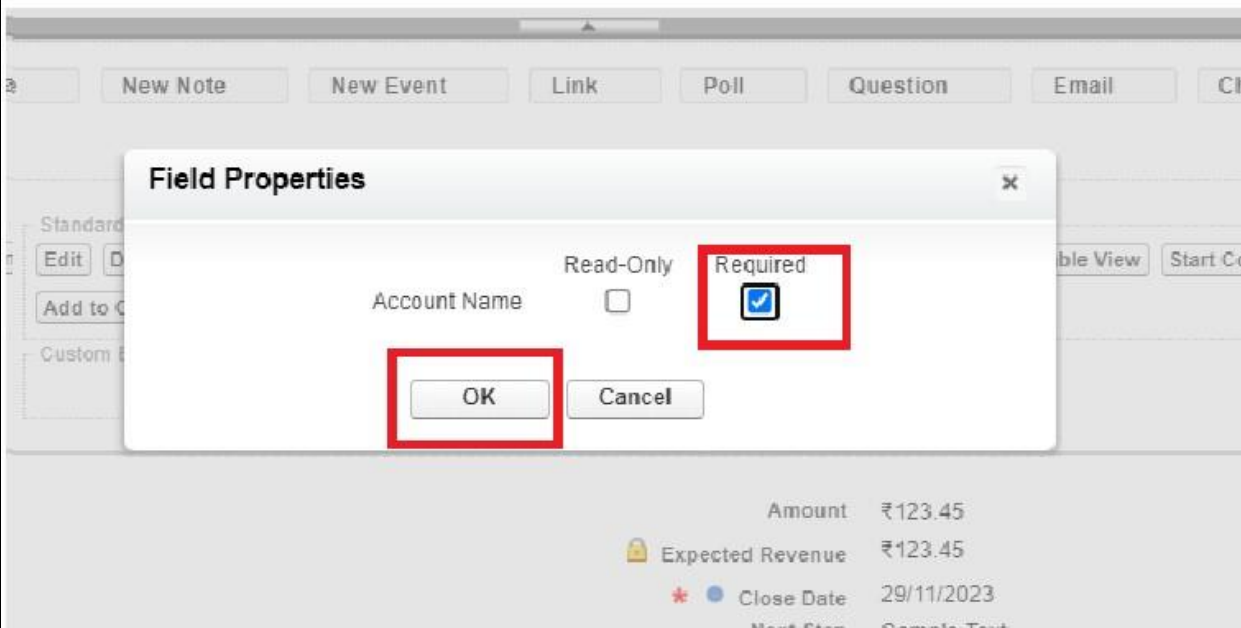
Page Layout in Salesforce allows us to customize the design and organize detail and edit pages of records in Salesforce. Page layouts can be used to control the appearance of fields, related lists, and custom links on standard and custom objects' detail and edit pages.

## 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.

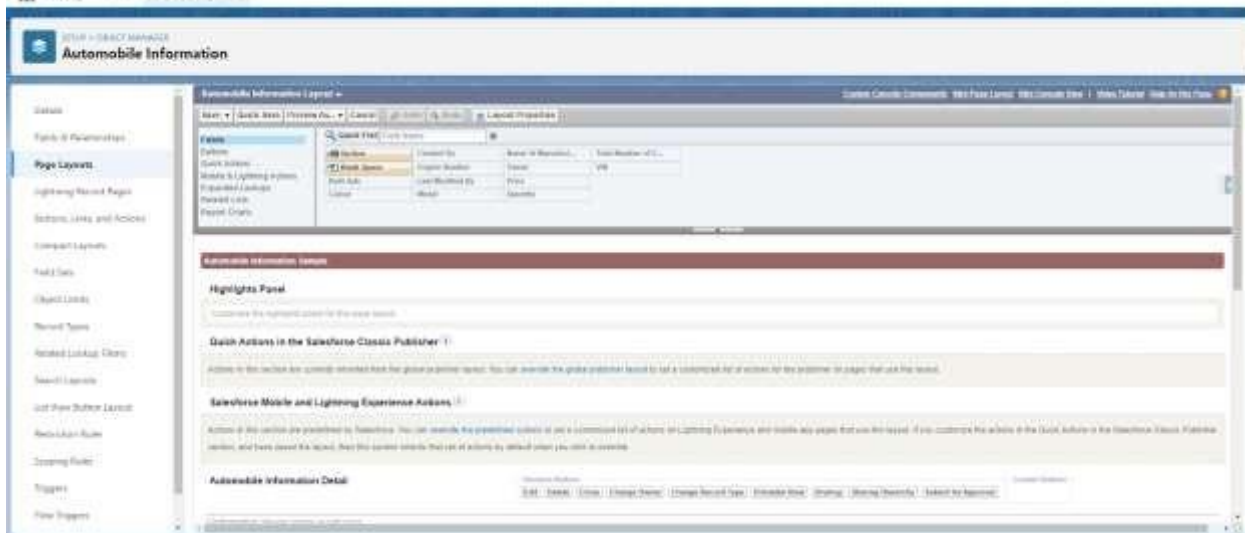


Step 4: check the Required box for Account name and click on Ok. Step 5: Click on Save.

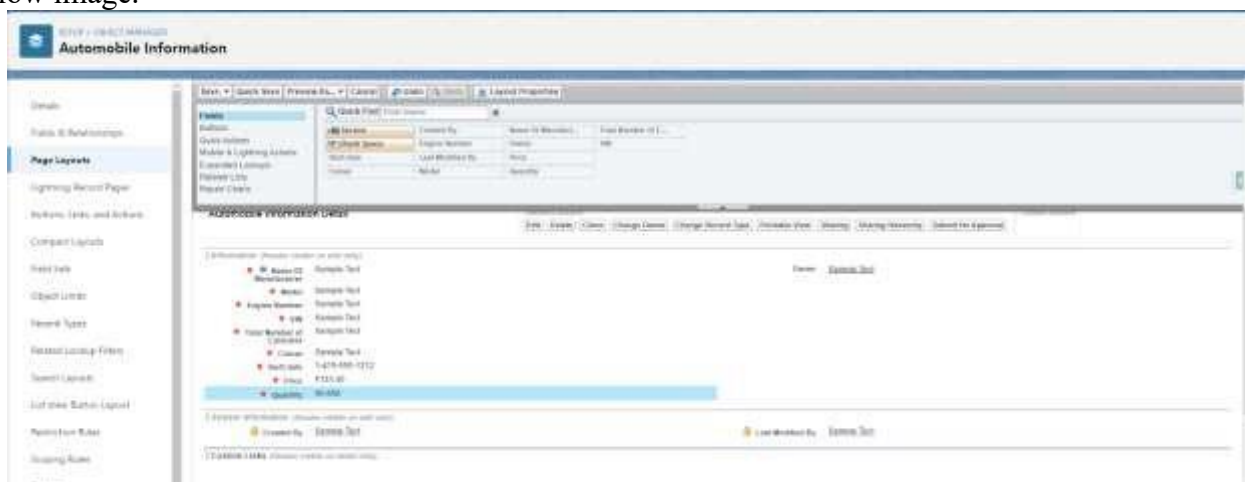
## Edit the Page layout for Automobiles Information

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

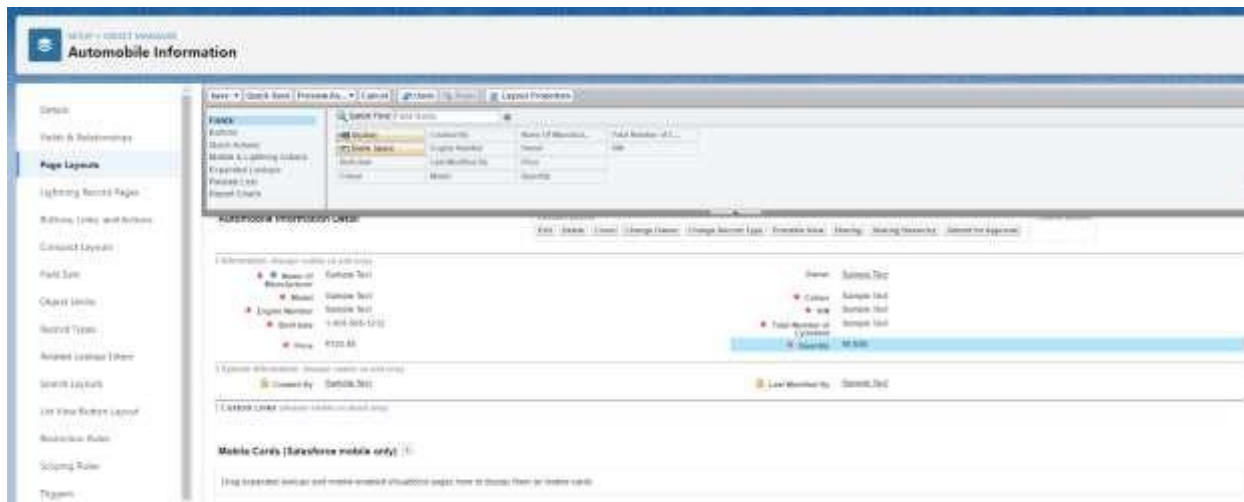
Step 2: Click on Page Layouts. Click on 'Automobile Information Layout'.



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__c>  
lstOpportunityAutomobile){  
        set<Id> AutomobileIds = new Set<Id>();  
        For(Opportunity__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__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__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__c> lstOpportunityAutomobile = [SELECT Unit_Price__c, Total_Price__c,
            Automobile__c, Quantity__c, Opportunity__c, Id FROM Opportunity__c WHERE Opportunity__c
            IN: oppIds];
        List<Invoice__c> lstInvoice = new List<Invoice__c>();
        For(Opportunity__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];
}
```

### 3}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:

EXTENSIONS MARKETPLACE

search: salesforce-extension-pack 2

3

Salesforce Extension Pack v55.5.1  
Extensions for developing on the Salesforce Platform  
Salesforce 9,47,031 ★★★★★ (43)  
Install 4

Details Change Log

Extension Pack (9)

- Salesforce CLI Integration  
Provides integration with the Salesforce CLI  
Salesforce Install
- Apex  
Provides code editing features for the Apex  
Salesforce Install
- Aura Components  
Provides code editing features for Aura Components  
Salesforce Install

codekiat.com

Extension: Salesforce Extension Pack X

Salesforce Extension Pack v56.5.1  
Salesforce 9,47,031 ★★★★★ (43)  
Extensions for developing on the Salesforce Platform  
Disable Uninstall ⚙️  
This extension is enabled globally.  
codekiat.com

Create a project in VS Code:

File Edit Selection View Go Run Terminal Help package.xml - devEdition - Visual Studio Code

EXPLORER

- DEVEDITION
  - .husky
  - .sfdx
  - .vscode
  - config
  - force-app
  - manifest
    - package.xml
  - scripts
  - eslintignore
  - forceignore
  - .gitignore
  - .prettiignore
  - .prettierrc
  - jest.config.js
  - package.json
  - README.md
  - sfdx-project.json

package.xml

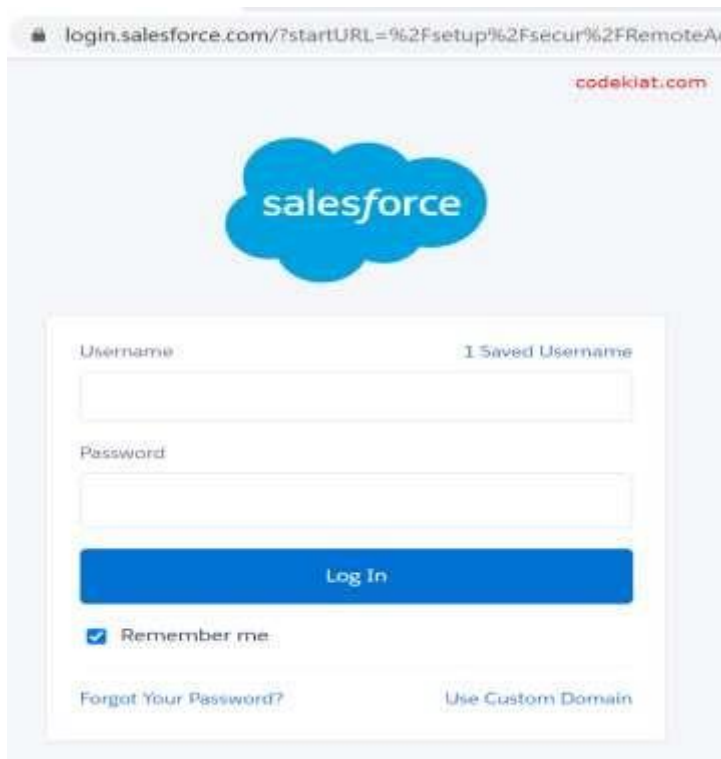
```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3   <types>
4     <members>*</members>
5     <name>ApexClass</name>
6   </types>
7   <types>
8     <members>*</members>
9     <name>ApexComponent</name>
10  </types>
11  <types>
12    <members>*</members>
13    <name>ApexPage</name>
14  </types>
15  <types>
16    <members>*</members>
17    <name>ApexTestSuite</name>
18  </types>
19  <types>
20    <members>*</members>
21    <name>ApexTestSuite</name>
22  </types>
23  <types>
24    <members>*</members>
25    <name>ApexTestSuite</name>
26  </types>
27  <types>
28    <members>*</members>
29    <name>ApexTestSuite</name>
30  </types>
31  <types>
32    <members>*</members>
33    <name>ApexTestSuite</name>
34  </types>
35  <types>
36    <members>*</members>
37    <name>ApexTestSuite</name>
38  </types>
39  <types>
40    <members>*</members>
41    <name>ApexTestSuite</name>
42  </types>
43  <types>
44    <members>*</members>
45    <name>ApexTestSuite</name>
46  </types>
47  <types>
48    <members>*</members>
49    <name>ApexTestSuite</name>
50  </types>
51  <types>
52    <members>*</members>
53    <name>ApexTestSuite</name>
54  </types>
55  <types>
56    <members>*</members>
57    <name>ApexTestSuite</name>
58  </types>
59  <types>
60    <members>*</members>
61    <name>ApexTestSuite</name>
62  </types>
63  <types>
64    <members>*</members>
65    <name>ApexTestSuite</name>
66  </types>
67  <types>
68    <members>*</members>
69    <name>ApexTestSuite</name>
70  </types>
71  <types>
72    <members>*</members>
73    <name>ApexTestSuite</name>
74  </types>
75  <types>
76    <members>*</members>
77    <name>ApexTestSuite</name>
78  </types>
79  <types>
80    <members>*</members>
81    <name>ApexTestSuite</name>
82  </types>
83  <types>
84    <members>*</members>
85    <name>ApexTestSuite</name>
86  </types>
87  <types>
88    <members>*</members>
89    <name>ApexTestSuite</name>
90  </types>
91  <types>
92    <members>*</members>
93    <name>ApexTestSuite</name>
94  </types>
95  <types>
96    <members>*</members>
97    <name>ApexTestSuite</name>
98  </types>
99  <types>
100   <members>*</members>
101   <name>ApexTestSuite</name>
102 </types>
103 </Package>

```

codekiat.com

## 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

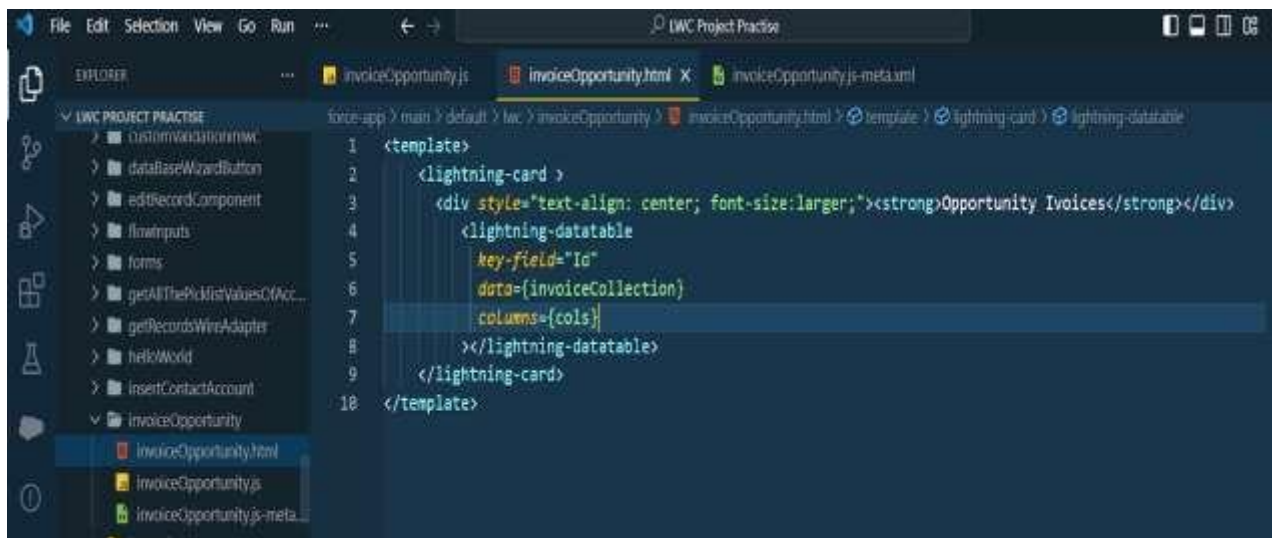
cols = [
  {label:"ID" , fieldName:'Name'},
  {label:"Opportunity Id" , fieldName:'Opportunity__c'},
  {label:"Quantity" , fieldName:'Quantity__c'},

  {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:

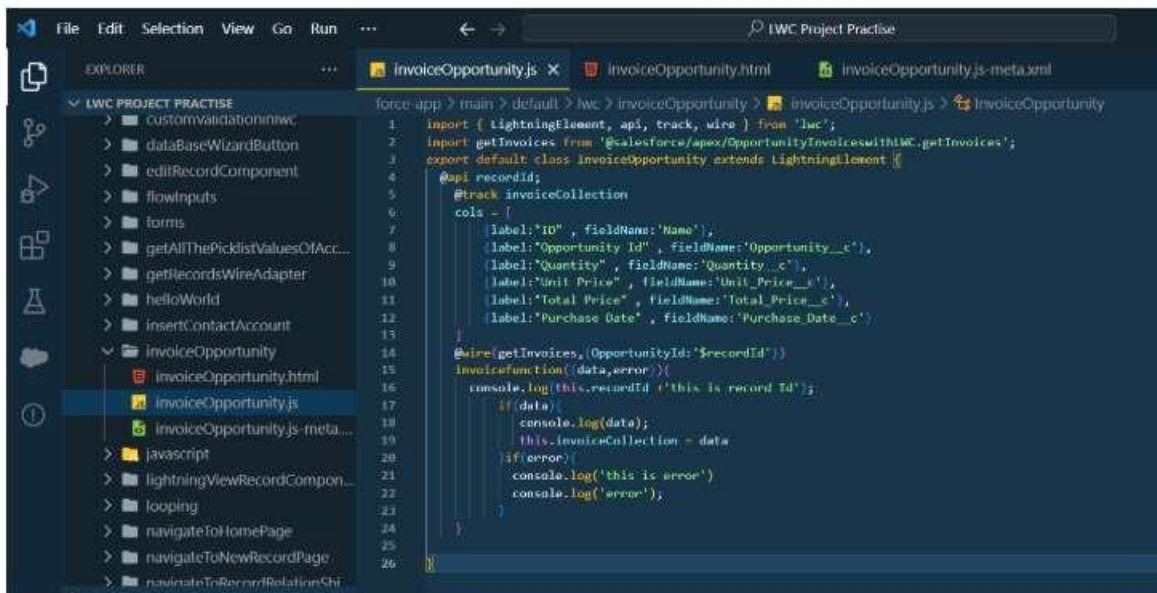
### JS File Code :

```

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');
    }
}
}
}

```





```

1 import { lightningElement, api, track, wire } from 'lwc';
2 import getInvoices from '@salesforce/apex/OpportunityInvoicesWithLWC.getInvoices';
3 export default class InvoiceOpportunity extends LightningElement {
4   @api recordId;
5   @track invoiceCollection;
6   cols = [
7     {label: 'ID', fieldName: 'Name'},
8     {label: 'Opportunity Id', fieldName: 'Opportunity__c'},
9     {label: 'Quantity', fieldName: 'Quantity__c'},
10    {label: 'Unit Price', fieldName: 'Unit_Price__c'},
11    {label: 'Total Price', fieldName: 'Total_Price__c'},
12    {label: 'Purchase Date', fieldName: 'Purchase_Date__c'}
13  ];
14  @wire(getInvoices, {OpportunityId: '$recordId'})
15  invoiceFunction(data, error) {
16    console.log(this.recordId, 'this is record Id');
17    if (data) {
18      console.log(data);
19      this.invoiceCollection = data;
20    }
21    if (error) {
22      console.log('this is error');
23      console.log('error');
24    }
25  }
26

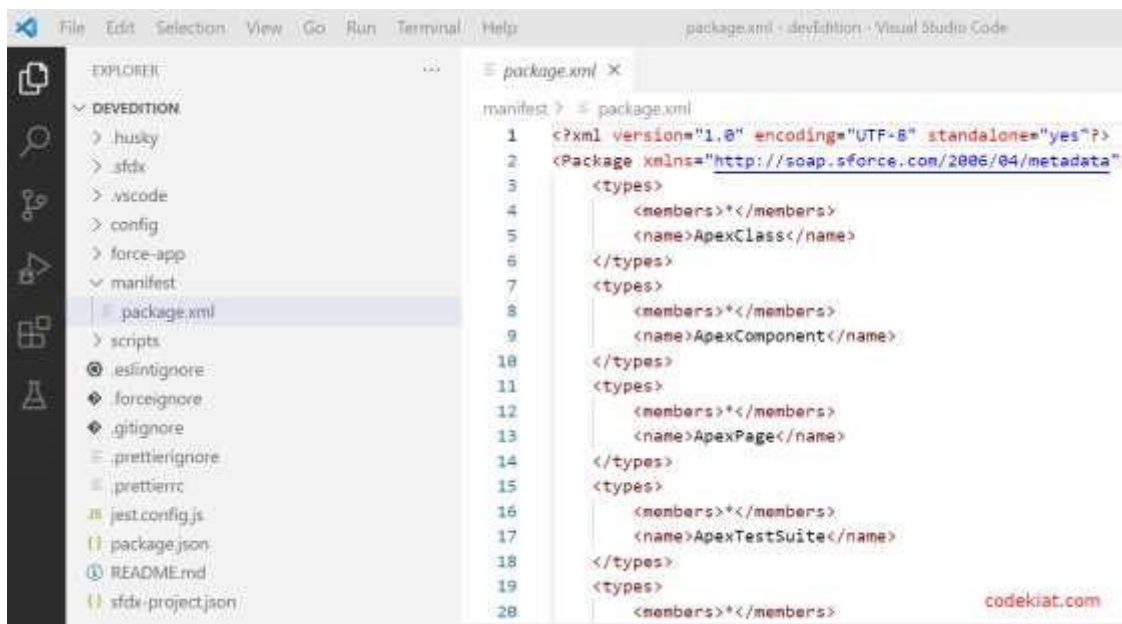
```

## Create Button to Add on Opportunity

Select the InvoiceOpportunity component

Label :- Invoices

Name :- Invoices



```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3   <types>
4     <members>*</members>
5     <name>ApexClass</name>
6   </types>
7   <types>
8     <members>*</members>
9     <name>ApexComponent</name>
10  </types>
11  <types>
12    <members>*</members>
13    <name>ApexPage</name>
14  </types>
15  <types>
16    <members>*</members>
17    <name>ApexTestSuite</name>
18  </types>
19  <types>
20    <members>*</members>
21  </types>
22

```



## 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 :



```
1 public class DeleteClosedLostOpportunities implements Schedulable{
2     public static void execute(SchedulableContext sc){
3         List<Opportunity> getLostOpportunities = [SELECT Id, Name From Opportunity Where StageName =: 'Closed Lost' LIMIT 50000];
4         if(!getLostOpportunities.isEmpty()){
5             Delete getLostOpportunities;
6         }
7     }
8 }
9
```

```
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.

Q apex classes

Custom Code

Apex Classes

Didn't find what you're looking for?  
Try using Global Search.


Click on Schedule Apex and enter the Job name. ○  
Job Name : DeleteOpportunitySchedule

### Schedule Apex

Schedule an Apex class that implements the 'Schedulable' interface to be automatically executed on a weekly or monthly interval.

Save Cancel

Job Name: DeleteOpportunitySchedule

Apex Class: DeleteClosedLostOpportun 

Schedule Apex Execution

Frequency: ☒ Weekly ☐ Monthly

Recurr every week on:

- ☐ Sunday
- ☒ Monday
- ☐ Tuesday
- ☐ Wednesday
- ☐ Thursday
- ☐ Friday
- ☐ Saturday

Start: 01/12/2023 { 01/12/2023 }

End: 01/01/2024 { 01/12/2023 }

Preferred Start Time: 10:00 am

Exact start time will depend on job queue activity.

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

Apex Classes ~ Salesforce - Developer Edition - Google Chrome

smartbridge328-dev-ed.develop.my.salesforce.com/\_ui/common/data/LookupPage?lkfm=edit...

### Lookup

DeleteClosedLostOpportun Go!

You can use "\*" as a wildcard next to other characters to improve your search results.

< Clear Search Results

#### Search Results

Name	Namespace Prefix	Api Version
DeleteClosedLostOpportunities		59

Copyright © 2000-2023 salesforce.com, inc. All rights reserved.

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:

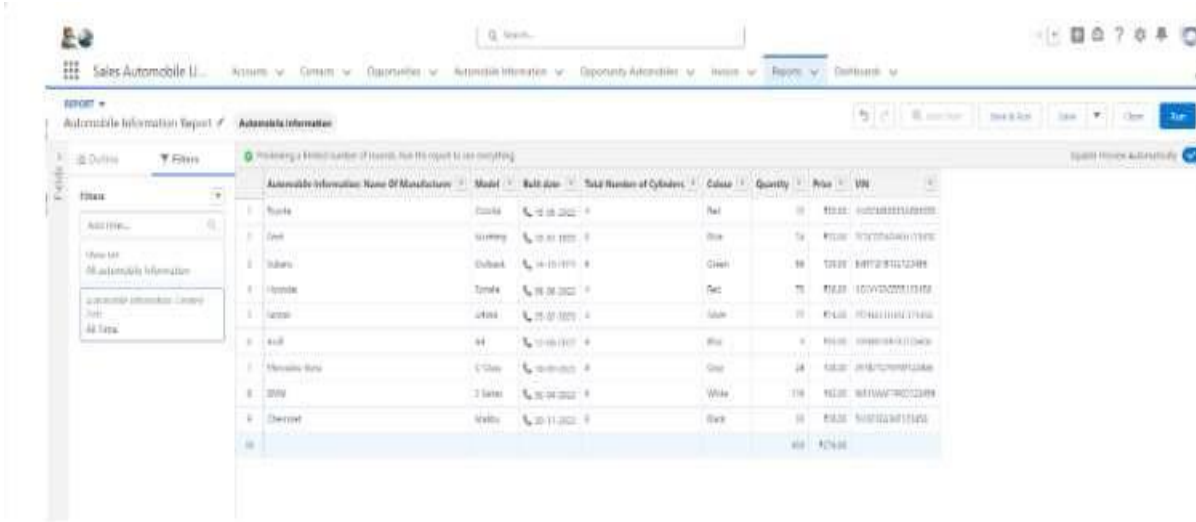
Opportunities

Previewing a limited number of records. Run the report to see everything.

Account Name	Opportunity Name	Owner Role	Opportunity Owner	Stage	Next Step	Lead Source	Type
NetScout Systems	NetScout	Account Executive	Manish Kumar	Closed Won		Web	Existing Customer - Upgrade
Subtotal							
Bullington Technologies Corp of America (1)	Bullington Technologies Consulting Plan Services	Account Executive	Manish Kumar	Closed Won		Web	New Customer
Subtotal							
Deutsche Telekom (1)	Deutsche Telekom Consulting	Account Executive	Manish Kumar	Qualification		Purchased List	New Customer
Subtotal							
Edge Communications (3)	Edge Emergency Services	Account Executive	Manish Kumar	Closed Won		Word of mouth	New Customer
	Edge Mobile	Account Executive	Manish Kumar	Closed Won		Word of mouth	Existing Customer - Upgrade
	Edge Web	Account Executive	Manish Kumar	Closed Won		Word of mouth	Existing Customer - Upgrade
	Edge Emergency Services	Account Executive	Manish Kumar	In Progress			Existing Customer - Replacement
Subtotal							
Global Health & Research LLC (5)	Global Health & Research Services	Account Executive	Manish Kumar	In Progress			Existing Customer - Upgrade
	Global Health & Research Consulting	Account Executive	Manish Kumar	Value Proposition		Employee Referral	Existing Customer - Upgrade
	Global Health & Research Consulting	Account Executive	Manish Kumar	Closed Won		Event at Referral	Existing Customer - Upgrade
	Global Health & Research	Account Executive	Manish Kumar	Closed Won		Event at Referral	Existing Customer - Upgrade
	Global Health & Research Consulting	Account Executive	Manish Kumar	Closed Won		Event at Referral	New Customer
Subtotal							

## Create Report on Automobile Information:

Filters:-



The screenshot shows a Salesforce Reports page for 'Sales Automobile U...'. The report is titled 'Automobile Information Report' and is currently in 'Preview' mode. The report displays a table of automobile information with columns: Automobile Information, Name Of Manufacturer, Model, Built Date, Total Number of Cylinders, Color, Quantity, Price, and VIN. The table contains 10 rows of data, including manufacturers like Honda, Opel, Subaru, Hyundai, Nissan, Audi, Mercedes-Benz, Volvo, and Chevrolet.

Automobile Information	Name Of Manufacturer	Model	Built Date	Total Number of Cylinders	Color	Quantity	Price	VIN
1	Honda	Civic	12-18-2022	4	Red	10	\$10,000	1000000000000000
2	Opel	Corvette	10-01-2020	8	Blue	50	\$10,000	1000000000000000
3	Subaru	Outback	14-10-1975	4	Green	50	\$10,000	1000000000000000
4	Hyundai	Territo	10-28-2022	4	Red	75	\$10,000	1000000000000000
5	Nissan	Altima	10-01-2020	4	Blue	10	\$10,000	1000000000000000
6	Audi	A4	10-01-2020	4	Blue	5	\$10,000	1000000000000000
7	Mercedes-Benz	C-Class	10-01-2020	4	Grey	20	\$10,000	1000000000000000
8	Volvo	S60	10-01-2020	4	White	10	\$10,000	1000000000000000
9	Chevrolet	Malibu	10-01-2020	4	Black	10	\$10,000	1000000000000000
10						100	\$10,000	

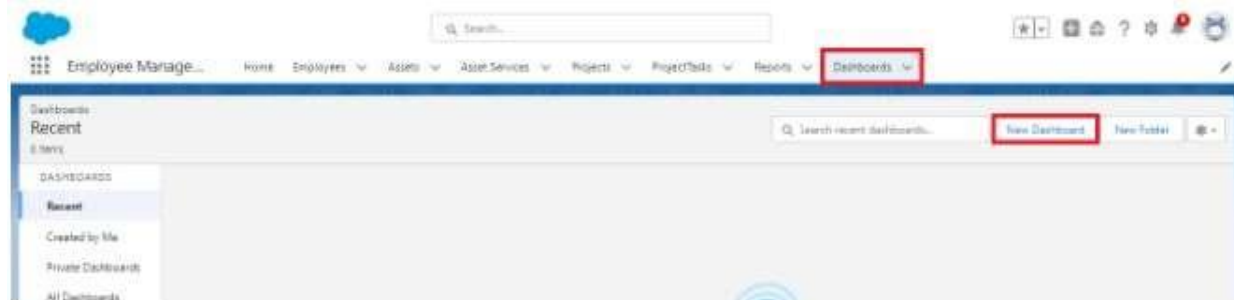
## Dashboard:

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.

## New Dashboard

\* Name

Dashboard 1

Description

Folder

Private Dashboards

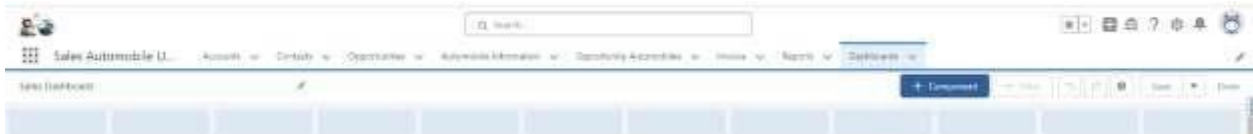
Select Folder

Cancel

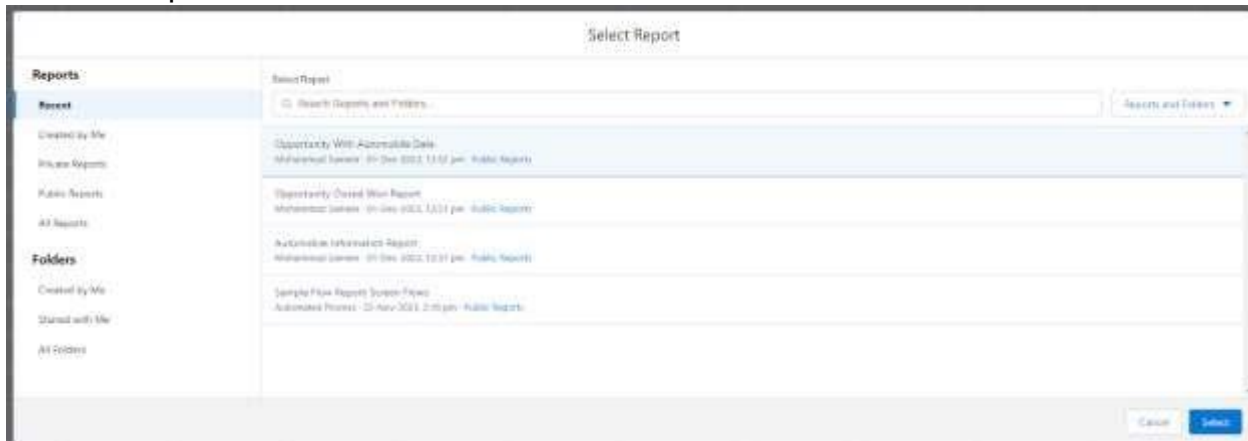
Create

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 aligns with the long-term goals of the automobile dealership, setting a foundation for future growth and digital transformation.