PROJECT DESCRIPTION

The Salesforce CRM implementation for automobile sales streamlines the entire sales process, enhancing efficiency and customer satisfaction. Through this system, sales teams can manage leads, track customer interactions, and automate follow-ups. It enables comprehensive customer profiling, allowing for personalized marketing strategies and targeted campaigns. The platform facilitates inventory management, ensuring real-time updates on available vehicles and their specifications. Integration with marketing tools enables seamless communication and lead nurturing. Additionally, the system provides insightful analytics, empowering decision-making by identifying sales trends and forecasting demand. Overall, the Salesforce CRM for automobile sales optimizes operations, fosters customer relationships, and drives revenue growth within the automotive industry.

PROJECT ABSTRACT

The implementation of Salesforce CRM in the automobile sales industry revolutionizes how sales teams engage with customers, manage inventory, and drive business growth. By leveraging Salesforce's robust features, this system streamlines lead management, tracks customer interactions, and automates followup processes, ensuring a seamless customer experience from initial contact to post sale support. The CRM enables personalized marketing by building comprehensive customer profiles, which support targeted campaigns and effective lead nurturing.

With real-time inventory management, the platform provides up-to-date details on vehicle availability and specifications, empowering sales teams to make informed decisions. Integrated marketing tools further enhance communication with leads, boosting conversion rates. Moreover, Salesforce CRM offers powerful analytics that helps identify sales trends, forecast demand, and uncover actionable insights, thereby improving strategic decision-making.

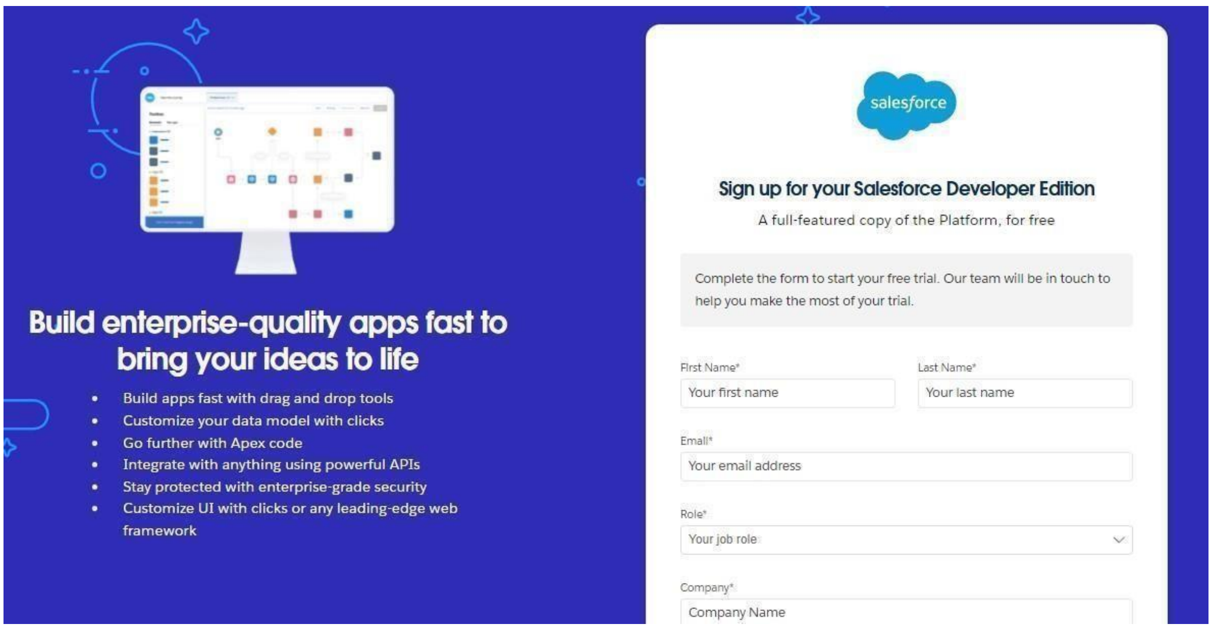
In summary, the Salesforce CRM for automobile sales optimizes internal operations, strengthens customer relationships, and accelerates revenue generation, offering a competitive edge in the dynamic automotive market.

TABLE OF CONTENTS

TASK 1-SALESFORCE DEVELOPER ACCOUNT CREATION:

Subtask 1.1: Creating Developer Account:

1. Go to https://developer.salesforce.com/signup
2. On the sign up form, enter the following details :

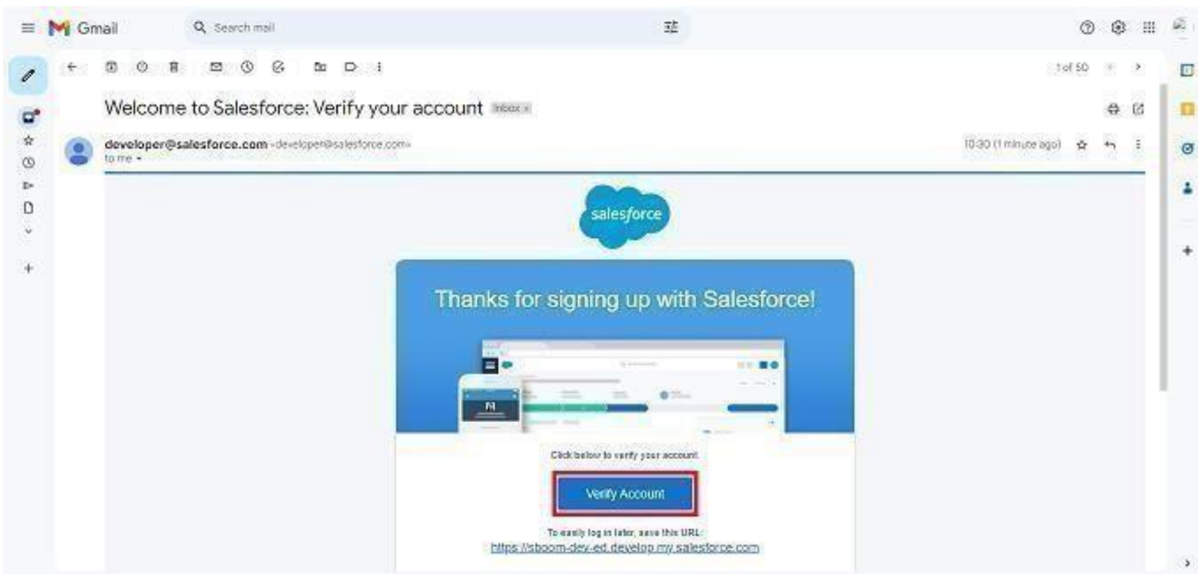


1. First name & Last name
2. Email
3. Role : Developer
4. Company : College Name
5. County : India
6. Postal Code : pin code
7. Username : should be a combination of your name and company

This need not be an actual email id, you can give anything in the format : username@organization.com Click on sign me up after filling these.

SUBTASK 1.2: Account Activation

1. Go to the inbox of the email that you used while signing up. Click on the verify account to activate your account. The email may take 5-10mins.



2. Click on Verify Account

3. Give a password and answer a security question and click on change password.

4. Then you will redirect to your salesforce setup page.

TASK 2- OBJECTS

SUBTASK 2.1: Create Automobile Information Object

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

2. Make sure to download the File into CSV format.

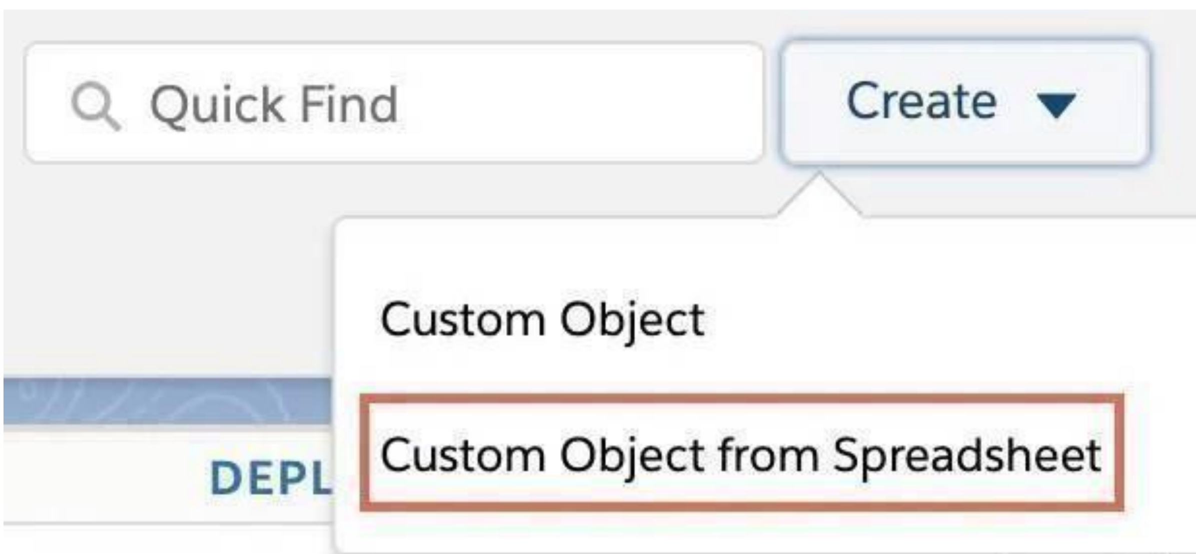
Note : Make sure to have the name of the file as “Automobile Information”.

Log into your salesforce account, click , then select Setup.

3. Click the Object Manager tab.

4. Click Create.

5. Select Custom Object from Spreadsheet.



6. Click Login With Salesforce.

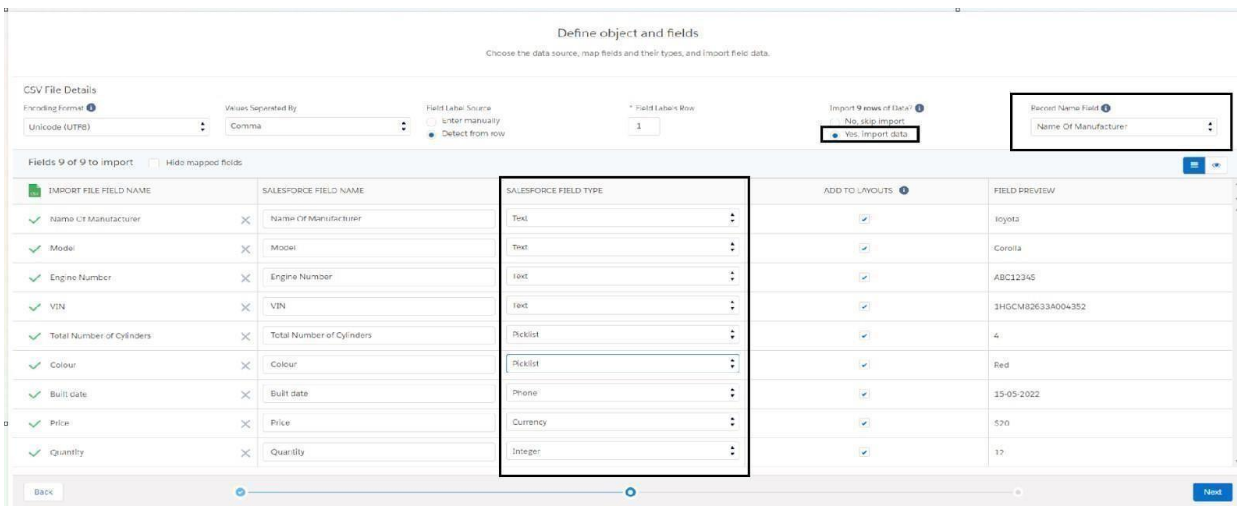
7. Enter your Salesforce account username and password. (which you have created in the Milestone 1, Activity 1)

8. Click Log In.

9. Click Allow.

10. Click Upload.

11. Navigate to the Automobile Information.csv file you downloaded and upload it. Salesforce automatically detects the fields and populates all its record data. Choose the Record Name field and make sure all fields are with the proper datatypes as below as they are.



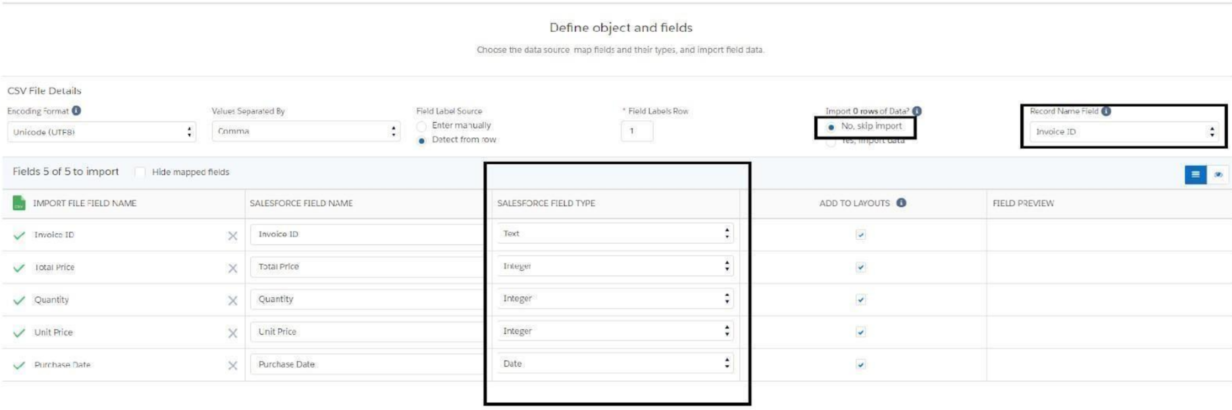
12. Click Next and enter the following settings.

13. Click Finish. The Automobile Information object is successfully created and data imported, all within minutes.

SUBTASK 2.2: Create Invoice Object

Create Invoice object, just as we have created an Automobile Information Object using [this sheet] Make sure to Download the File into CSV Format.

Note: Make sure you do field mapping with proper field type as shown below.



SUBTASK 2.3: Create Automobile Object

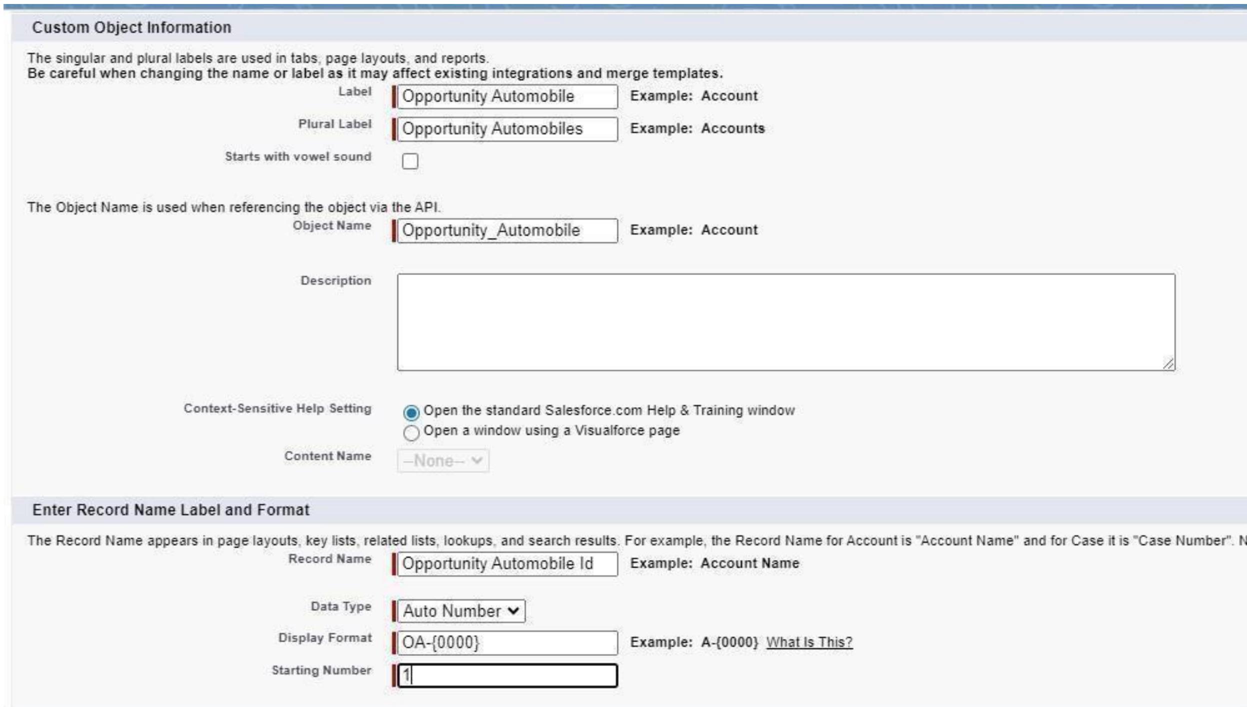
The purpose of creating an Automobile custom object is to store and manage information about Invoice.

To create an object:

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.

1. Enter the label name>> Opportunity Automobile
2. Plural label name>>Opportunity Automobiles
3. Enter Record Name Label and Format
   - Record Name >> Opportunity Automobile Id
   - Data Type >> Auto Number

- Display Format >> OA-{0000}
- Starting Number >> 1



2. Click on Allow reports.

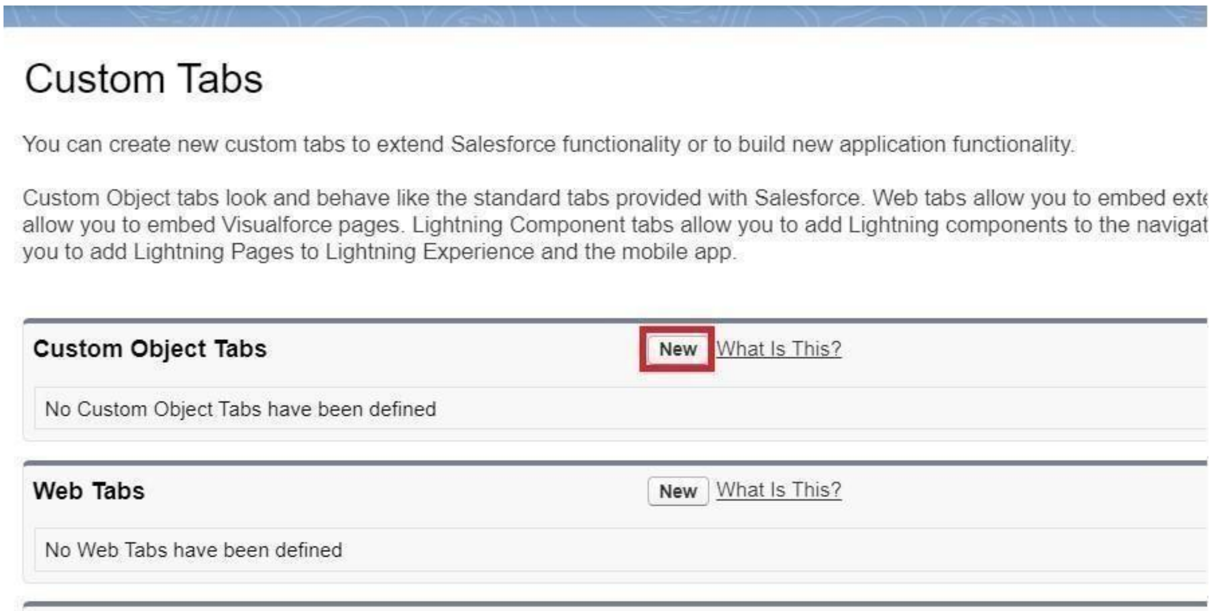
3. Allow search

4. Save.

TASK 3- TABS

SUBTASK 3.1: Creating a Custom Tab

1. Go to setup page >> type Tabs in Quick Find bar >> click on tabs >> New (under custom object tab)

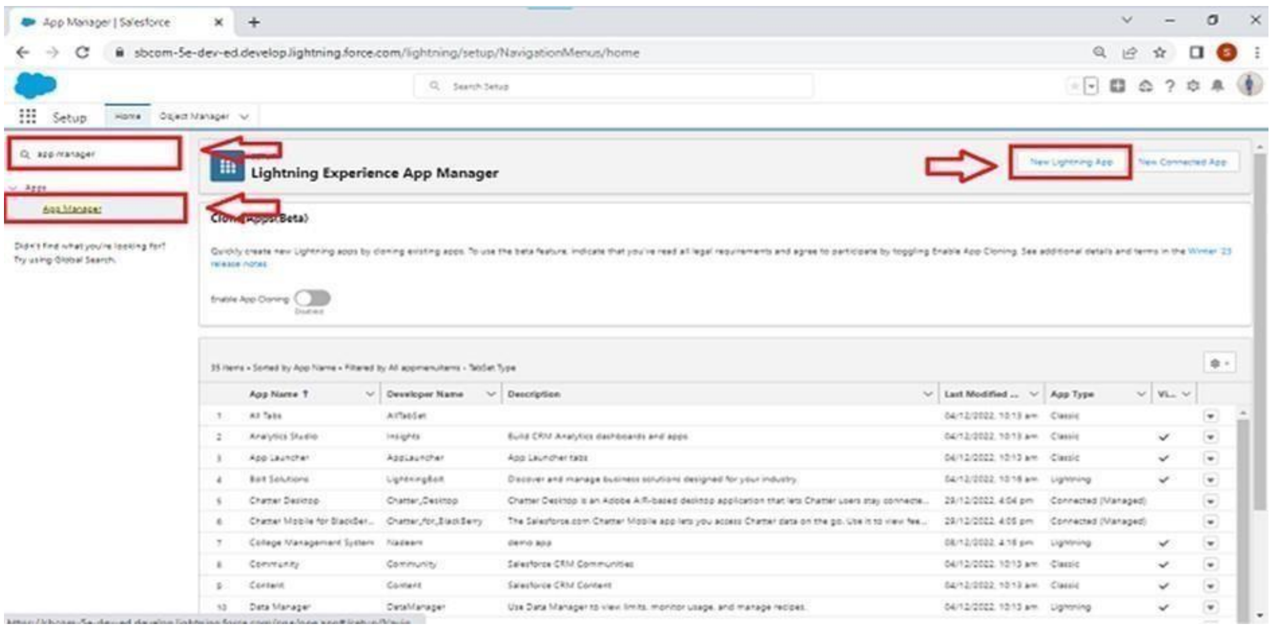


2. Select Object(Opportunity Automobile) >> Select any tab style >> Next (Add to profiles page) keep it as default >> Next (Add to Custom App) keep it as default >> Save.

Note: Tabs for Automobile Information & Invoice objects do get created automatically. We do not need to create tabs for those objects.

TASK 4- THE LIGHTNING APP

SUBTASK 4.1: Create a Lightning App

1. Go to setup page >> search “app manager” in quick find >> select “app manager” >> click on New lightning App.
2. 

3. 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

   Then click Next >> (App option page) keep it as default >> Next >> (Utility Items) keep it as default >> Next.

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.

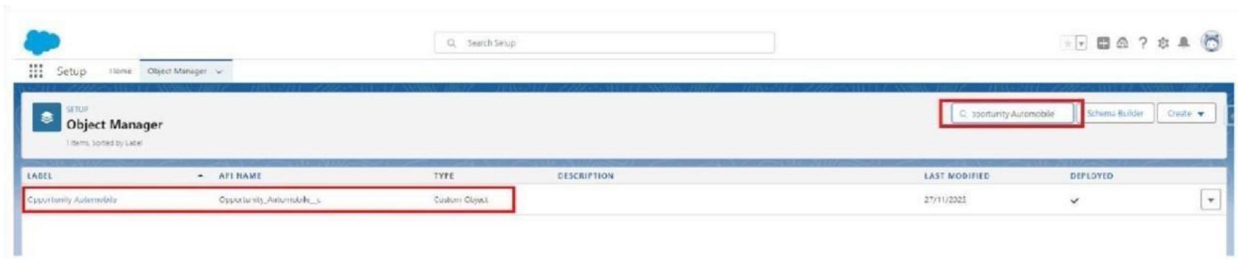
5. Search profiles (System administrator) in the search bar >> click on the arrow button >> save & finish.

TASK5- FIELDS AND RELATIONSHIPS

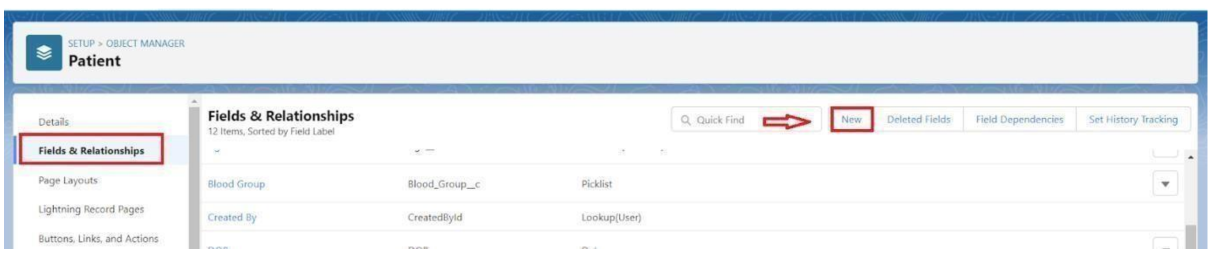
SUBTASK 5.1: Creating Opportunity Master Detail Relationship Field in Opportunity Automobile Object

To create fields in an object:

1. Go to setup >> click on Object Manager >> type object name(Opportunity Automobile) in quick find bar>> click on the object.



2. Now click on "Fields & Relationships" >> New



3. Select Data type as "Master Details Relationship".

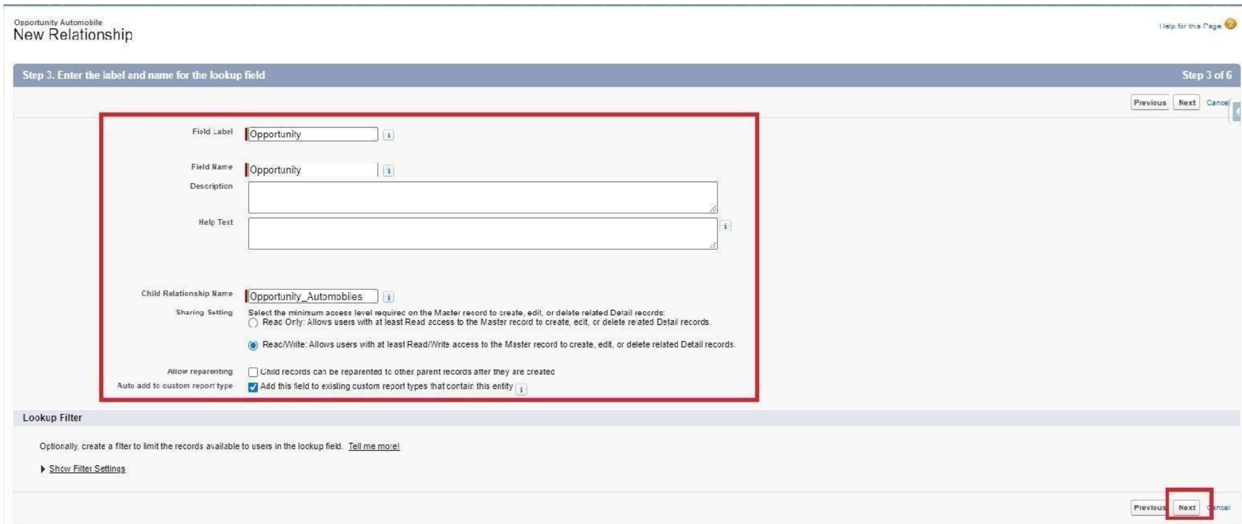
4. Click on Next

5. Fill the above as following:

• Field Label: gets auto Generated(Opportunity) • Field

Name : gets auto generated(Opportunity) • Click on Next

>> Next >> Save and new.

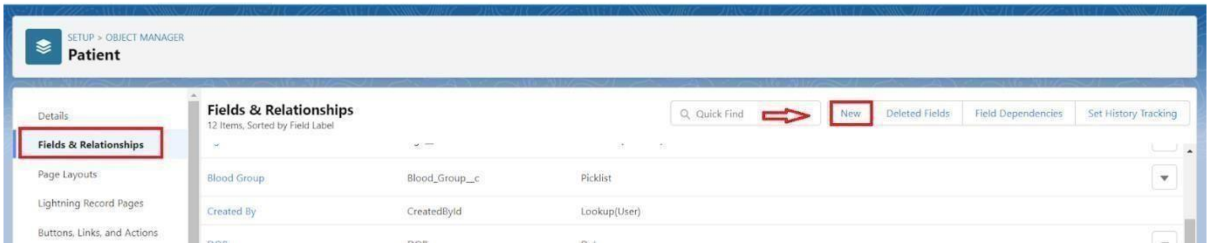


SUBTASK 5.2: Creating the AutoMobile Information Lookup Field in Opportunity Automobile Object

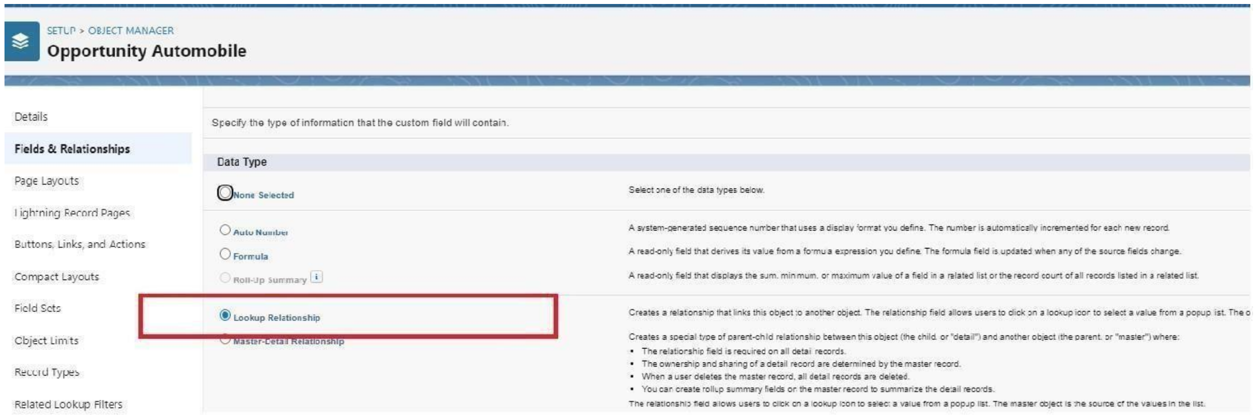
To create fields in an object:

1. Go to setup >> click on Object Manager >> type object name(Opportunity Automobile) in quick find bar>> click on the object.

2. Now click on "Fields & Relationships" >> New



3. Select Data type as “Lookup RelationShip”.



4. Click on Next

5. Fill the above as following:

a Field Label: Automobile

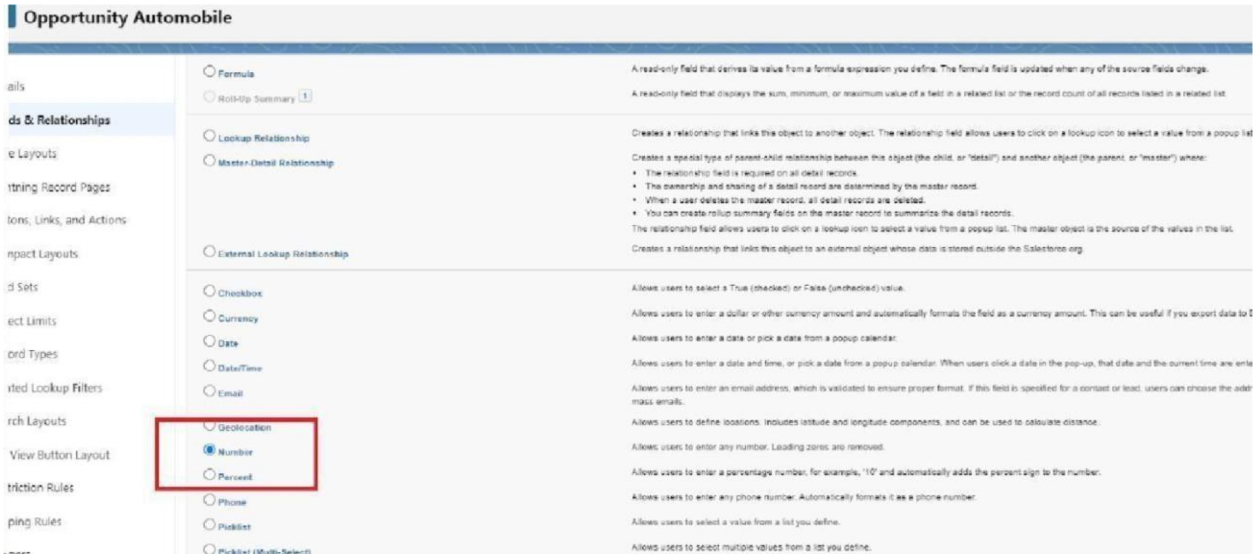
b Field Name : Automobile

Click on Next >> Next>> Save and new

SUBTASK 5.3: Creating Quantity Number Field in Opportunity Automobile Object

To create fields in an object:

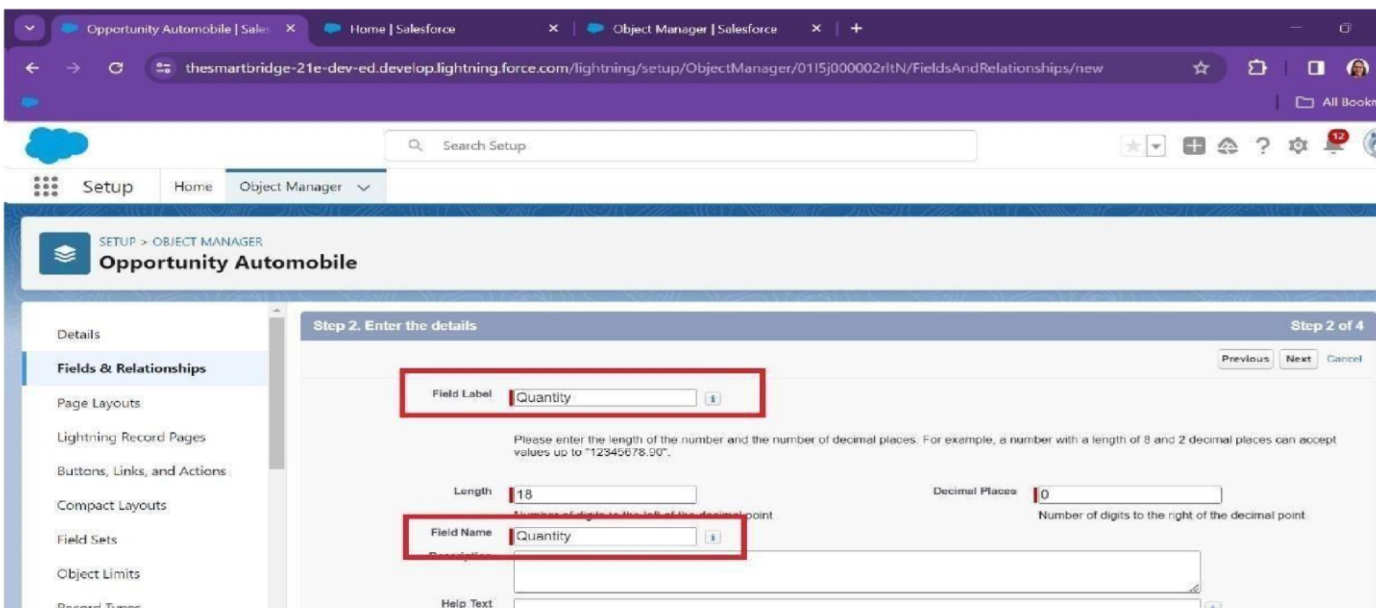
1. Go to setup >> click on Object Manager >> type object name(Opportunity Automobile) in quick find bar >> click on the object.
2. Now click on “Fields & Relationships” >> New.
3. Select Data type as “Number” and click Next.



a. Field Label >> Quantity

b. Field Name >> Quantity

Check that Required Check box.



Click Next >> Next >> Save & New.

SUBTASK5.4: Creating Formula Field in Opportunity Automobile Object

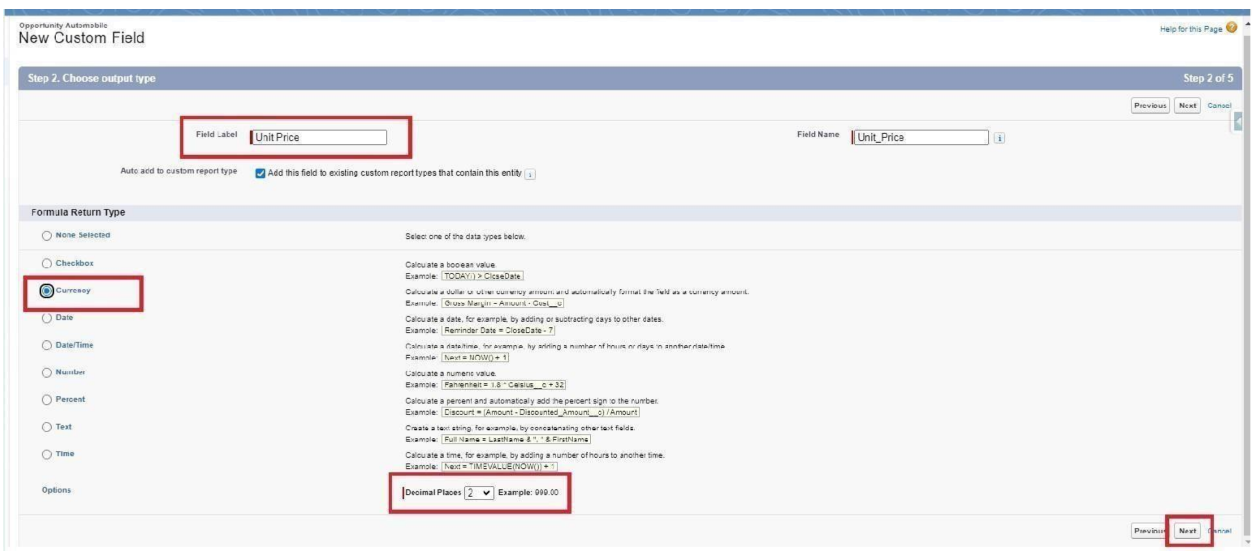
To create fields in an object:

1. Go to setup >> click on Object Manager >> type object name(Opportunity Automobile) in quick find bar >> click on the object.

1. Now click on “Fields & Relationships” >> New.

2. Select Data type as “Formula” and click Next.

3. Give Field Label and Field Name as “Unit Price” and select formula return type as “Currency” and change the decimal values to two and click next.



4. Under Advanced Formula write down the formula : Automobile_r.Price__c

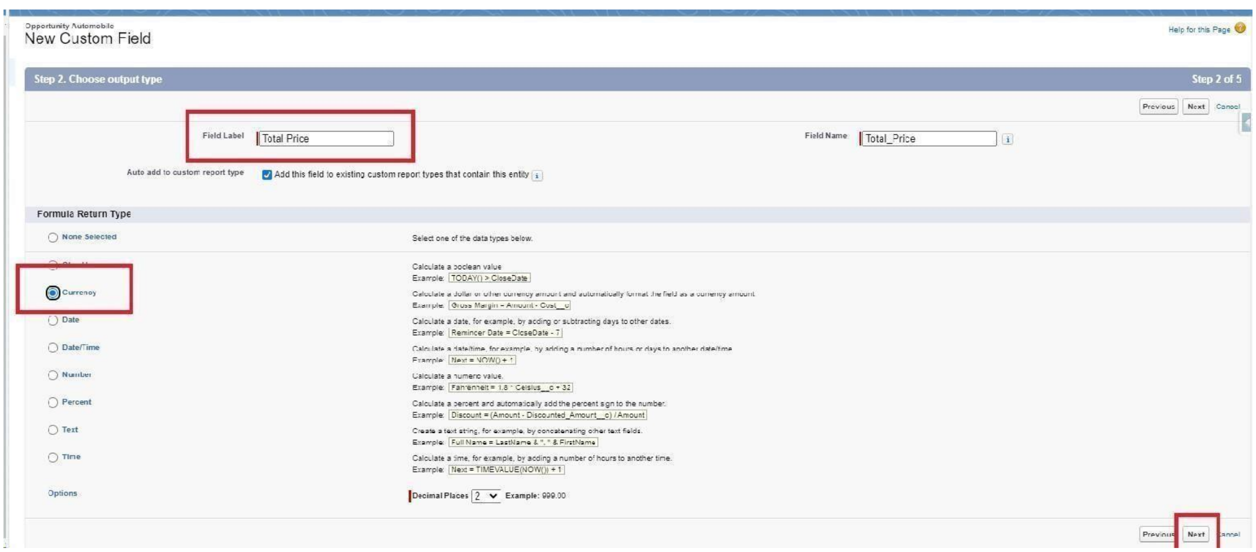
5. click “Check Syntax” and Next >> Next >> Save & New.

SUBTASK 5.5:Creating the Formula field in Opportunity Automobile Object

To create fields in an object:

1. Go to setup >> click on Object Manager >> type object name(Opportunity Automobile) in quick find bar >> click on the object.

1. Now click on “Fields & Relationships” >> New.
2. Select Data type as “Formula” and click Next.
3. Give Field Label and Field Name as “Total Price” and select formula return type as “Currency” and change the decimal values to two and click next.



4. Under Advanced Formula write down the formula : Unit_Price__c * Quantity__c
5. click “Check Syntax” and Next >> Next >> Save.

SUBTASK 5.6: Updating field in Invoice Object

To Update fields in an object:

1. Go to setup ? click on Object Manager ? type object name(Invoice) in quick find bar? click on the object.
2. Now click on "Fields & Relationships" , Click on the edit of Invoice Id field.

3.Select Data type as "Auto Number " and click Next.

   a. Display Format :- I-{0000}
   b. StartingNumber:- Click Save.

SUBTASK 5.7: Creating Remaining Fields in Objects

Now create the remaining fields using the data types mentioned.

- Field Name : Opportunity
- Data type : Master Detail relationship
- Object : Opportunity

TASK 6-PAGE LAYOUTS

SUBTASK 6.1: 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.

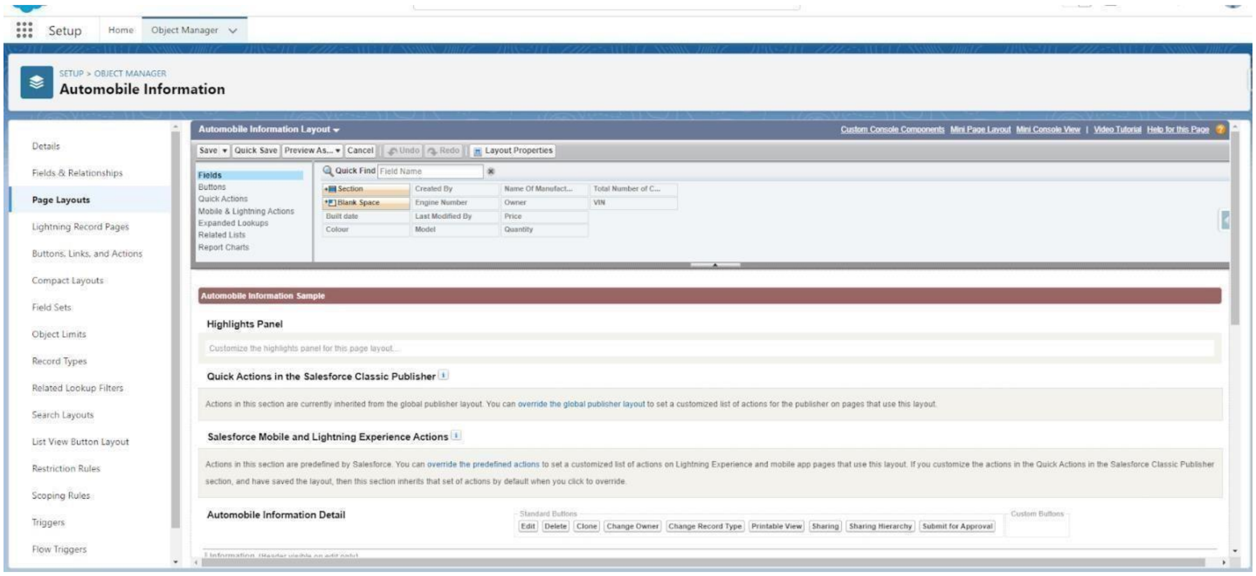
Step 5: Click on Save.

SUBTASK 6.2: 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.

TASKS 7- APEX TRIGGERS

SUBTASK 7.1: Opportunity Automobile quantity

Use Case : Whenever Opportunity Closed won Than Neglect / Minus the Quantity From Automobile Information on the Bases of Opportunity Automobile quantity.

1. Login to the respective trailhead 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 “Opportunity Handler Class ”.

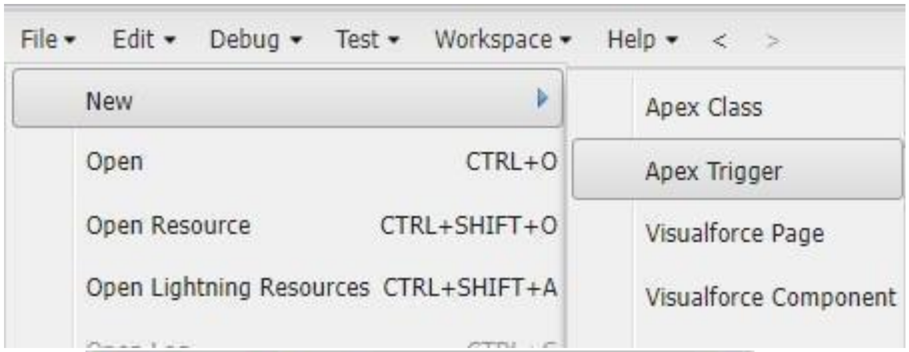
```
public class OpportunityHandlerClass {

    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);
            }
        }
```

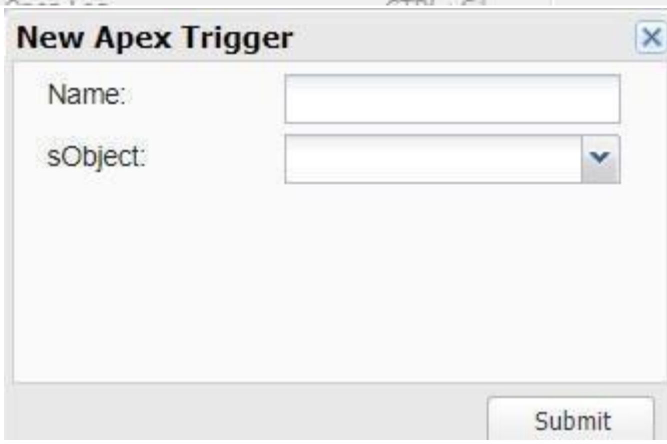
```
        set<Id> opportunityIds = new set<Id>();
        for(Opportunity opp : LstOpportunity){
            if(opp.StageName =='Closed Won' ){
                opportunityIds.add(opp.Id);
            }
        }
        Map<Id,Opportunity_Automobile__c> lstOpportunityAutomobile =new Map<Id,Opportunity_Automobile__c>([SELECT Id, Opportunity__c, Automobile__c, Quantity__c, Unit_Price__c, Total_Price
                                                                                                           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 :

How to create a new trigger :

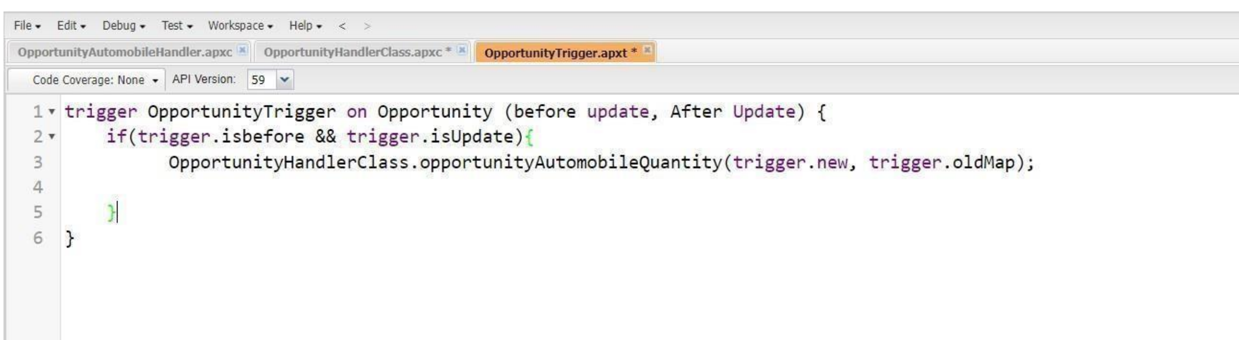
1. While still in the account, navigate to the gear icon in the top right corner.
2. Click on developer console and you will be navigated to a new console window.
3. Click on the File menu in the toolbar, and click on new? Trigger.
4. Enter the trigger name and the object to be triggered.
5. Name : OpportunityTrigger
6. sObject : Opportunity





In this project , trigger is called whenever the particular records sum exceed the threshold i.e minimum business requirement value. Then the code in the trigger will get executed.

1. Trigger for Opportunity Object.



SUBTASK 7.2: Opportunity-Automobile Error

Use Case : If Quantity of Automobile is Zero or Less than The Quantity from The Opportunity-Automobile Than Throw an error .

Login to the respective trailhead account and navigate to the gear icon in the top right corner.

1. Click on the Developer console. Now you will see a new console window.
2. In the toolbar, you can see FILE. Click on it and navigate to new and create

   New apex class.

3. Name the class as “Opportunity Automobile Handler ”.

```
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 ' + lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c );
            }
        }
    }

```

Trigger Handler :

How to create a new trigger :

1. While still in the trailhead account, navigate to the gear icon in the top right corner.

2. Click on developer console and you will be navigated to a new console window.

3. Click on the File menu in the toolbar, and click on new? Trigger.

4. Enter the trigger name and the object to be triggered.

5. Name : Opportunity Auto Mobile Trigger

6. Subject : Opportunity _Automobile __c

Trigger :

Handler for the Opportunity _Automobile __c Object

OpportunityAutomobileHandler.apxc | OpportunityHandlerClass.apxc | OpportunityTrigger.apxt | OpportunityAutoMobileTrigger.apxt

Code Coverage: None | API Version: 59

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

SUBTASK 7.3: Invoice Creation Trigger

Use Case : Whenever an opportunity is Closed won then create the Invoice on the Bases of Opportunity Automobile Data.

Login to the respective trailhead account and navigate to the gear icon in the top right corner.

1. Click on the Developer console. Now you will see a new console window.

2.In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.

3.Name the class as “InvoiceCreation”.

```
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 :

For this class we don't need to create any trigger, we will call this Code in "Opportunity Trigger".

1. Go on files and click on open.
2. Click on triggers.
3. Double click on Opportunity Trigger.


```
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);
    }
}
```

SUBTASK 7.4: Check contact role

Use Case : Whenever an opportunity is Going to Closed won then check it has the

contact role or Not.

Login to the respective trailhead account and navigate to the gear icon in the top right corner.

1. Click on the Developer console. Now you will see a new console window.
2. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
3. Name the class as "ContactRoleCheck".

```
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 :

For this class we don't need to create any trigger, we will call this Code in "Opportunity Trigger".

- Go on files and click on open.
- Click on triggers.
- Double click on Opportunity Trigger.

TriggerCode: :

```
Code Coverage: None    API Version: 59
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);
    }
}
```

TASK 8- LWC COMPONENT

SUBTASK 8.1: Create Apex Class to Get Invoices

1. Login to the respective account and navigate to the gear icon in the top right corner.

2. Click on the Developer console.

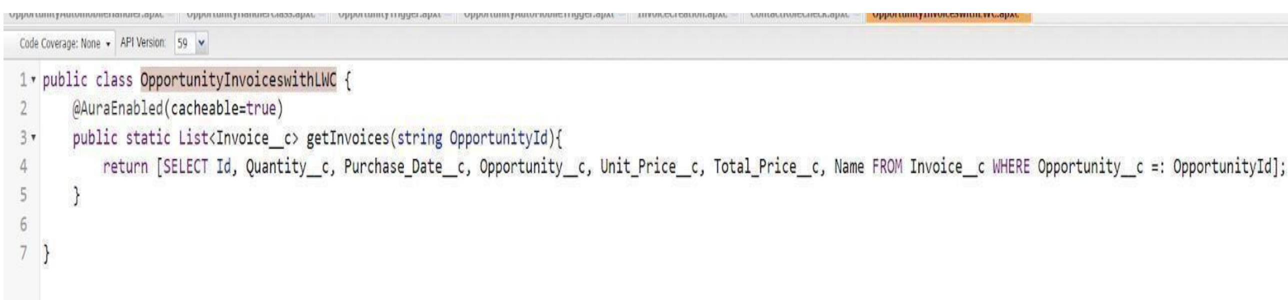
3. Now you will see a new console window.

In the toolbar, you can see FILE. Install Salesforce CLI

4.

5. Click on it and navigate to new and create New apex class.

6. Name the class as "OpportunityInvoiceswithLWC".



SUBTASK 8.2: Install Salesforce CLI

The Salesforce CLI is a powerful command line interface that simplifies development and build automation when working with your Salesforce org.

[Download and install Salesforce CLI]

To confirm that the Salesforce CLI is installed and working correctly, you can open a command prompt and type sfdx. This will display the version number of the Salesforce CLI that is currently installed on your system.

```
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
```

SUBTASK 8.3: 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.

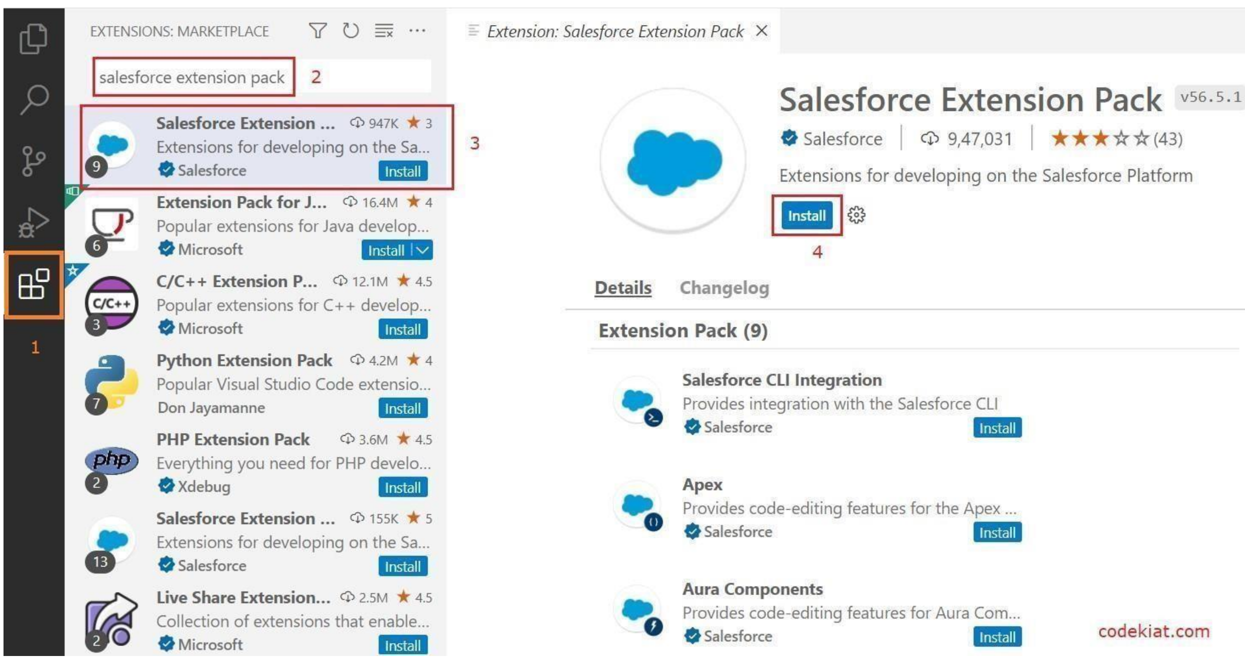
[Download the version of the software]() that is compatible with your operating system and install it.

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

SUBTASK 8.4: Install the Salesforce Extension Pack

In the VS Code,

1. go to extensins (1) as shown in the image below.
2. Search with the Salesforce extension pack (2) as shown in the image below.
3. select Salesforce Extension Pack from the list (3) as shown in the image below.
4. Click the Install button (4) as shown in the image below.

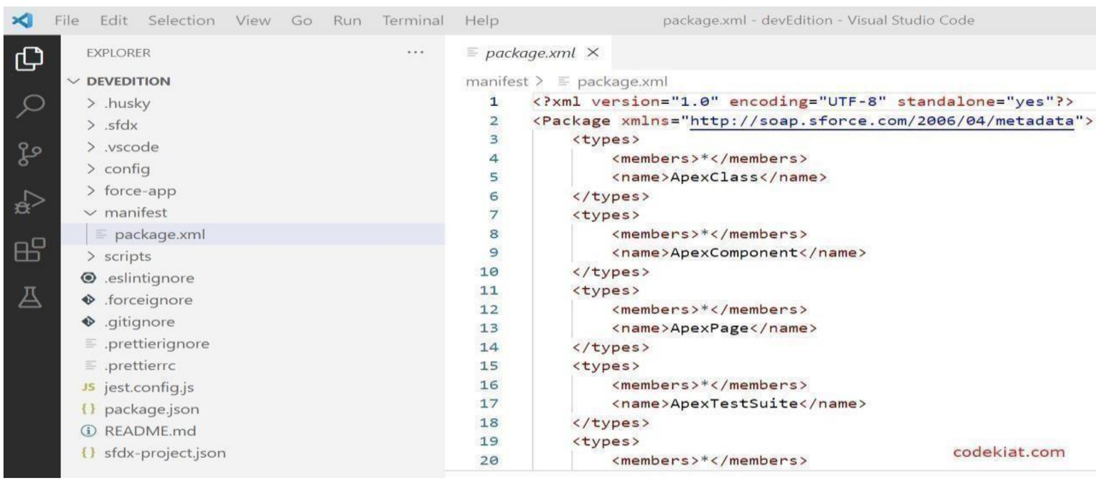


The extension pack is installed successfully

Install the Salesforce Extension Pack

SUBTASK 8.5: Create a project in VS Code

1. Press CTRL + SHIFT + P, type sfdx: create
2. select SFDX: Create Project with Manifest
3. Select the Standard project template
4. Type a project name and Click Enter.
5. Select the folder (create a new folder if required) and click Create Project
6. The new project is created with package.xml



SUBTASK 8.6: Authorize an org

Establish a connection between the local project and the Salesforce instance to retrieve and deploy the components.

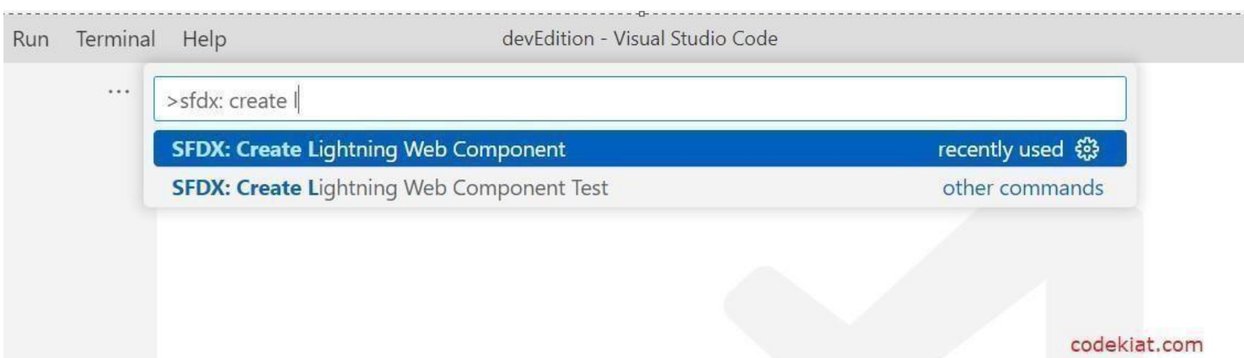
- Press CTRL + SHIFT + P, type sfdx: authorize.
- select SFDX: Authorize an Org from the list

- Choose your Salesforce instance.
- For developer edition and production instances select Production.
- For this demonstration, I used the developer edition, hence it is Production.
- Give a project name and press Enter
- The Salesforce login page opens in the browser. • Enter the credentials and click Log In
- It will be successfully authorized.

SUBTASK 8.7: Create Lightning Web Component

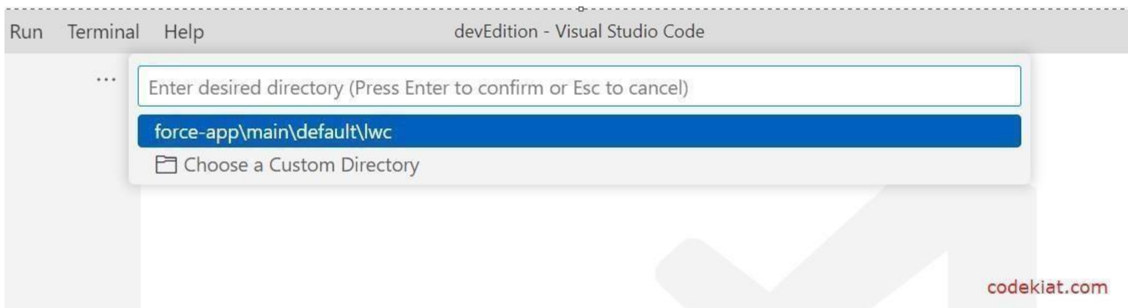
XML File :

1. In the VS Code, press CTRL + SHIFT + P, type sfdx: create lightning in the search bar, and select SFDX: Create Lightning Web Component

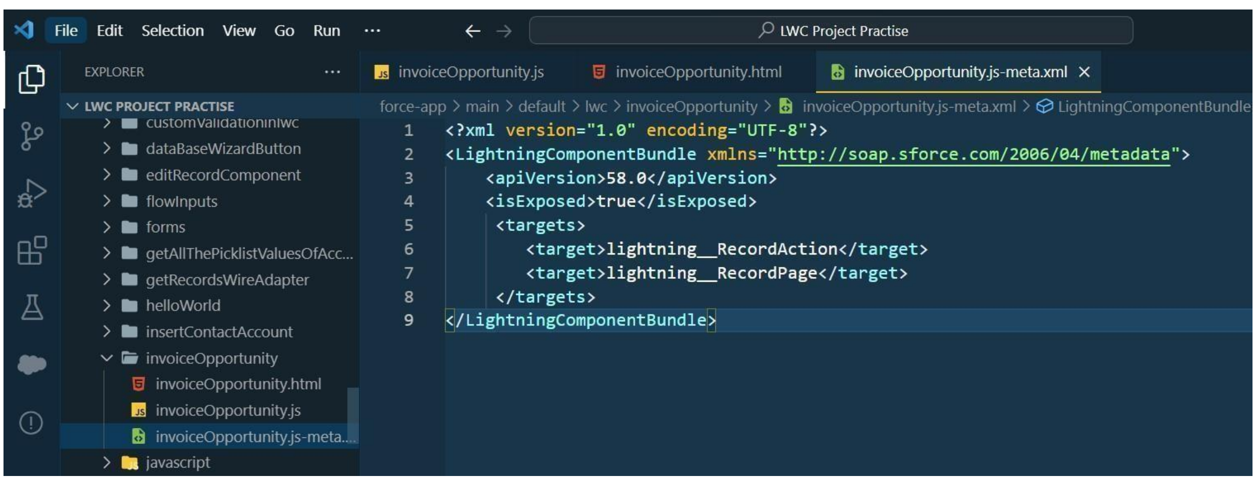


2. Give the name “InvoiceOpportunity” and press Enter.

3. Choose the directory.

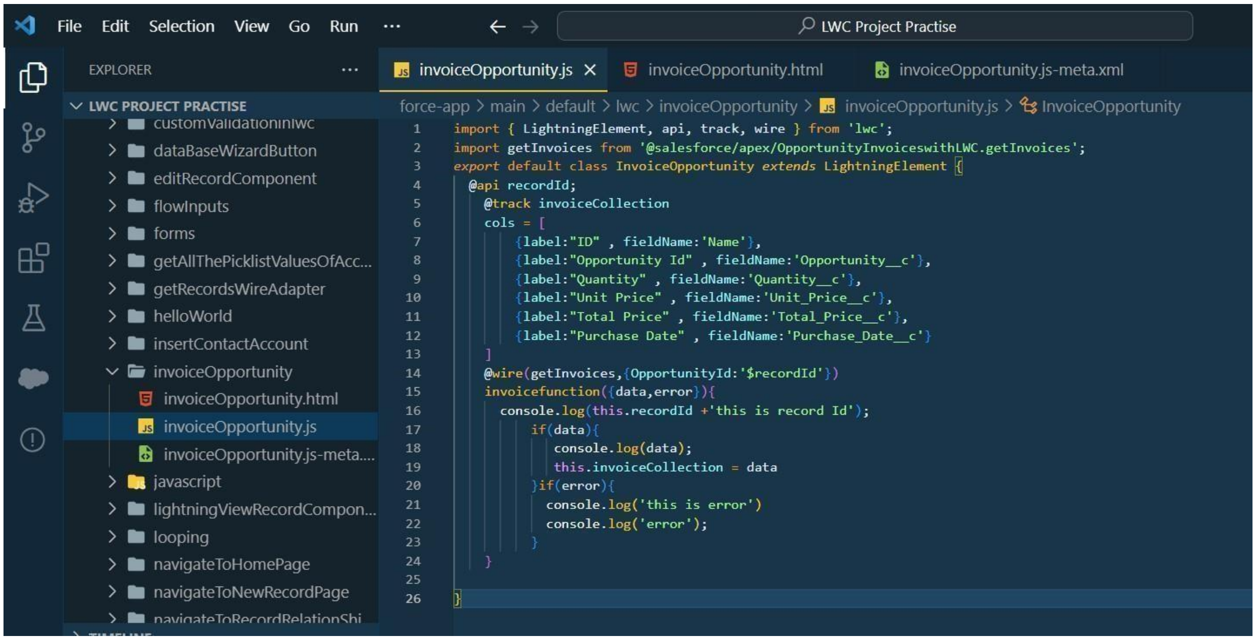


4. LWC is created successfully.



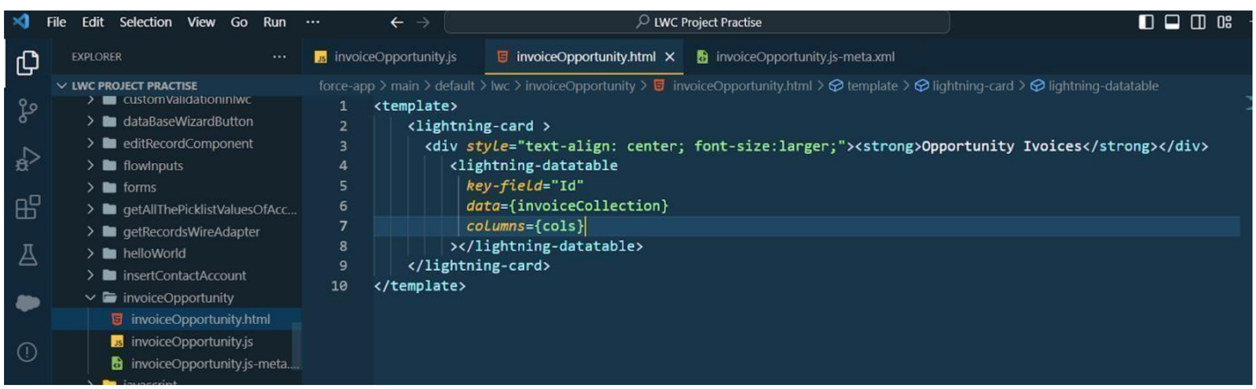
JS File :

1. Copy and paste the below-mentioned code in the InvoiceOpportunity.js and update the apiVersion tag with the latest API version.



HTML File :

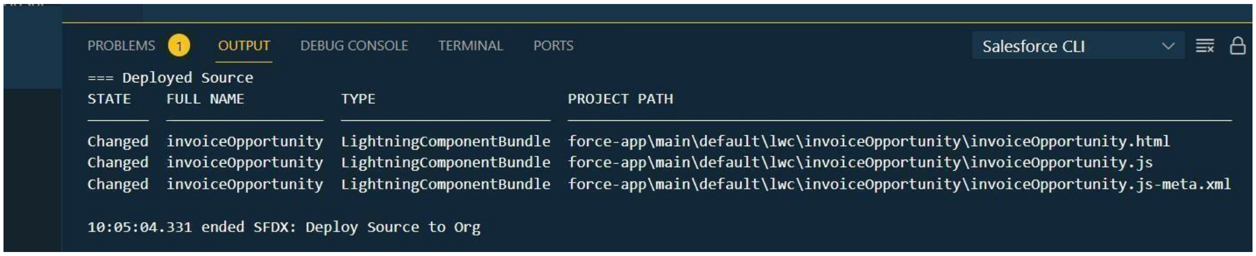
1. Copy and paste the below-mentioned code in the InvoiceOpportunity.html and update the apiVersion tag with the latest API version.



Deploy Component:

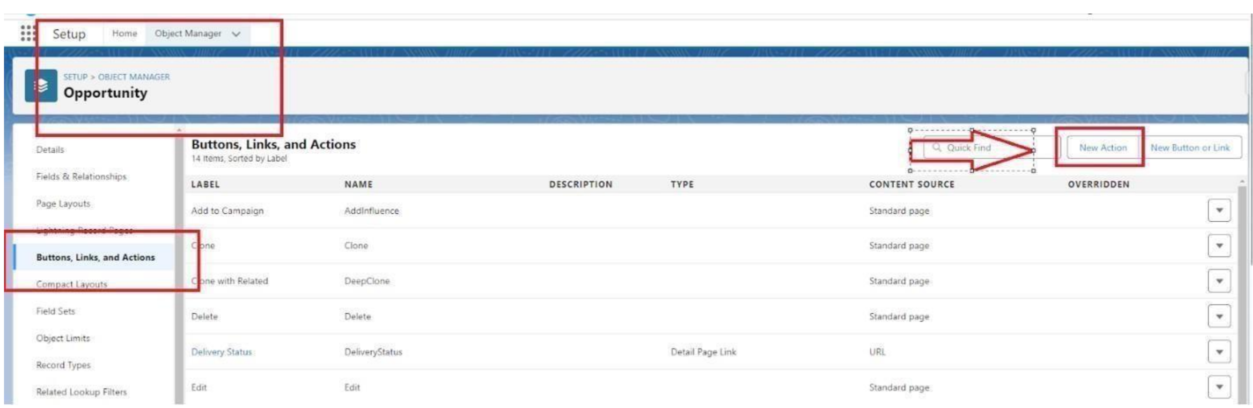
1. Right-click on the component folder, and select SFDX: Deploy Source to Org to deploy the component to the org.

2. Once the deployment is complete, you will see the below-highlighted message in the output tab



SUBTASK8.8: Create Button to Add on Opportunity

1. To add the newly created component to the view, Go to Salesforce Setup
2. Click on Object Manager
3. Search Opportunity and Click on it .
4. click on Button Links and Action.
5. click on the New Action.



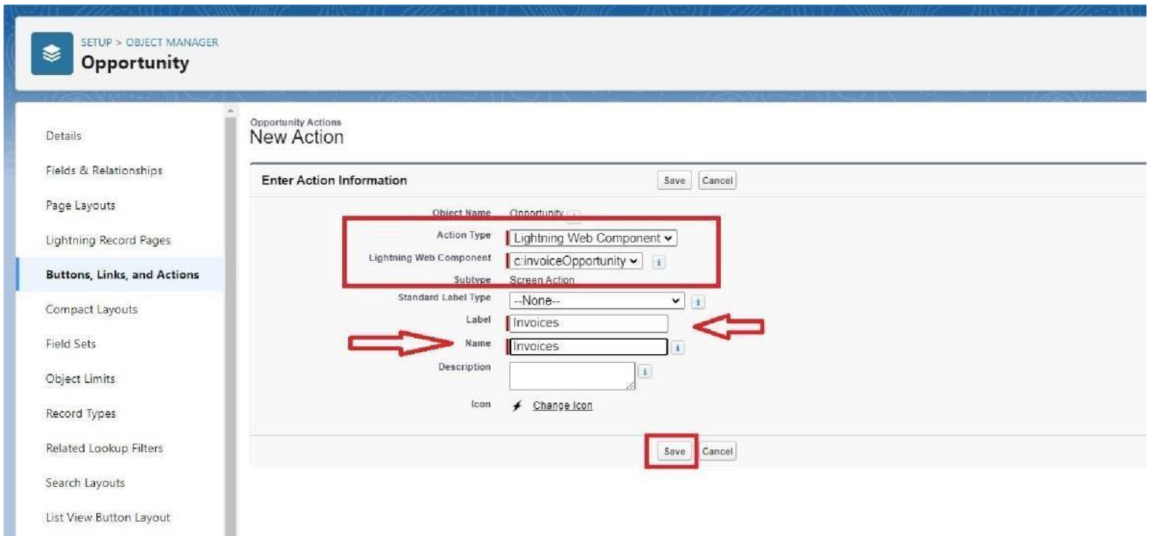
6. Select Action type as Lightning Web Component

7. Select the InvoiceOpportunity component

   a. Label :- Invoices

   b. Name :- Invoices

As given on below image



Click on Save and your action Button is Ready.

SUBTASK 8.9: Add Invoice Opportunity into Opportunity Record Page

1. On Opportunity Object Manager Click on Page layout.
2. Click on OpportunityLayout.
3. Click on Mobile And Lightning Action as show on below Image
4. Search for invoice on Quick Find
5. Drag and Drop the Invoice into Salesforce Mobile and Lightning Experience Actions. Click on Save.

TASK 9- APEX SCHEDULERS

SUBTASK 9.1: 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.

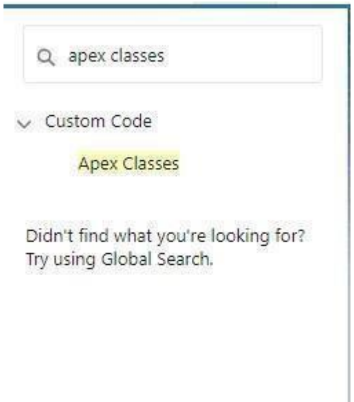
4. Name the class as “DeleteClosedLostOpportunities ”

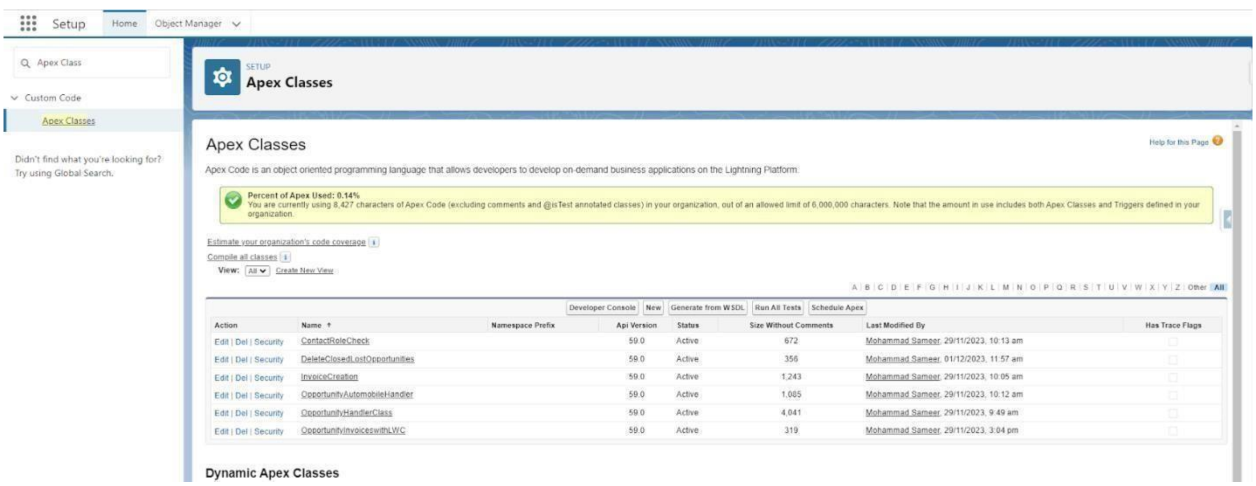
CODE SNIPPET :

```
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;
        }
    }
}
```

Schedule the Apex class:

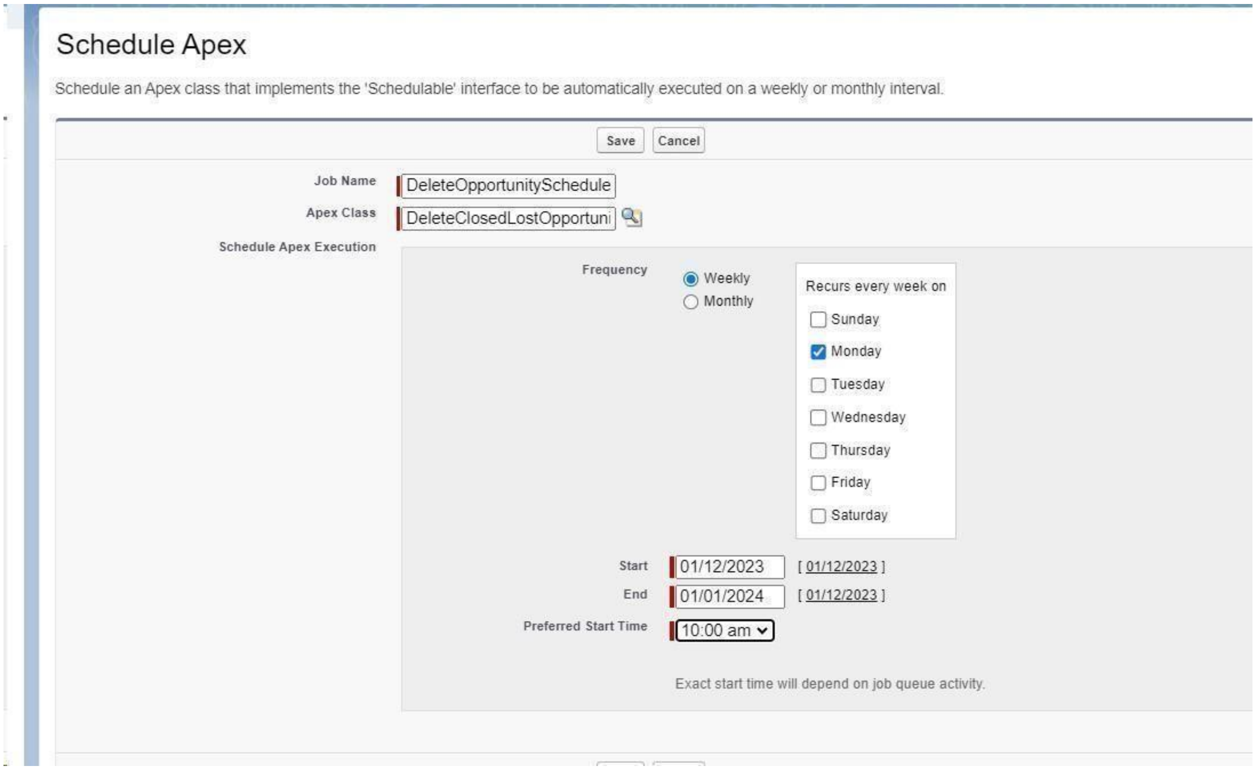
- Go to the Home page in your salesforce account.
- In the search bar, enter Apex and click on Apex Classes.





- Click on Schedule Apex and enter the Job name.

o Job Name : DeleteOpportunitySchedule

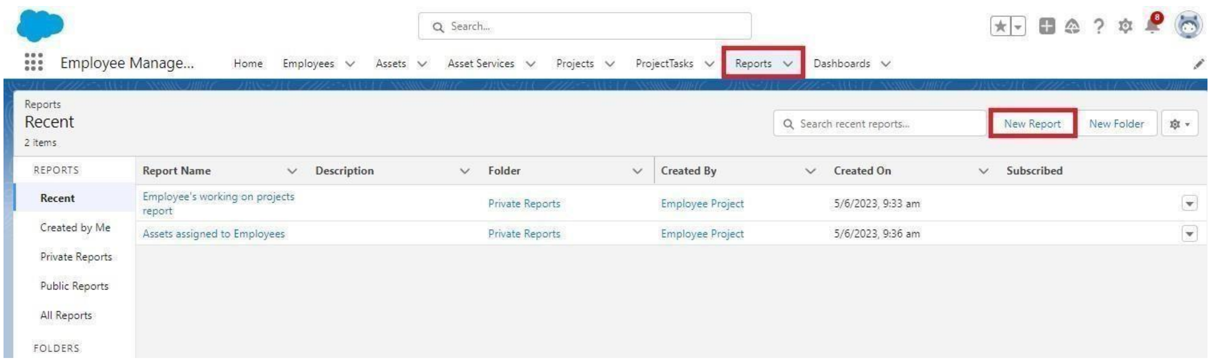


- Now click on the search icon present near the Apex class : Goto the Lookup icon beside ? click on it ? select DeleteClosedLostOpportunities.
- In the Schedule Apex section , select weekly and select Monday mentioned and preferred time as 10:00 AM.
- Click on Save. Now enter Apex in the search box and select Apex jobs.
- You can see that the batch job is in queue and will run whenever the day mentioned comes.

TASKS 10- REPORTS

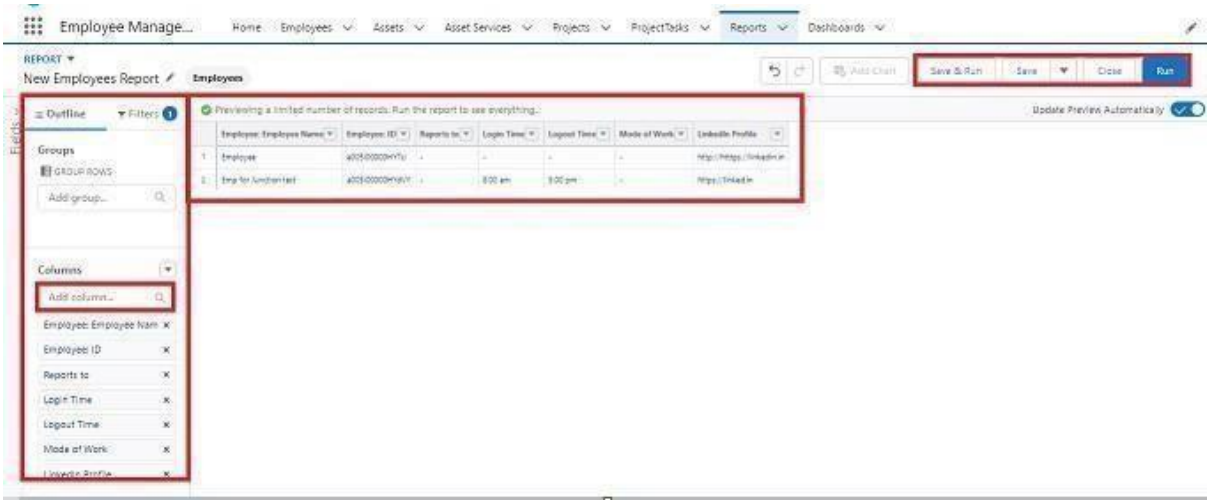
SUBTASK 10.1: Create Report on Opportunity 1. Go to the app >> click on the reports tab 2.

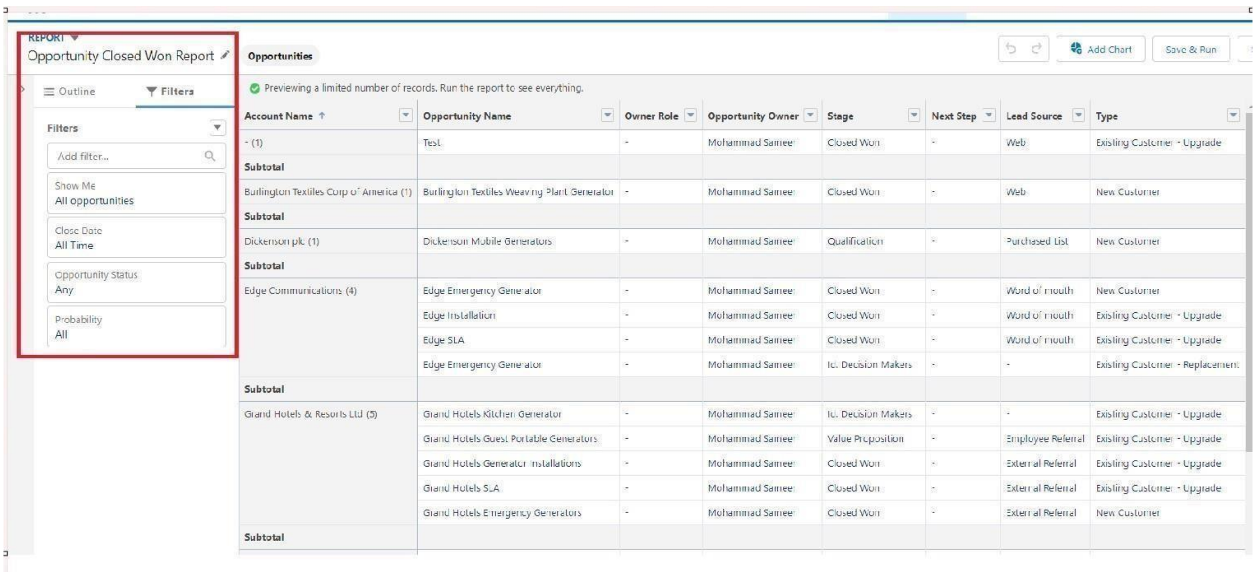
Click New Report.



2. Customize your report

- Add fields from left pane as shown below





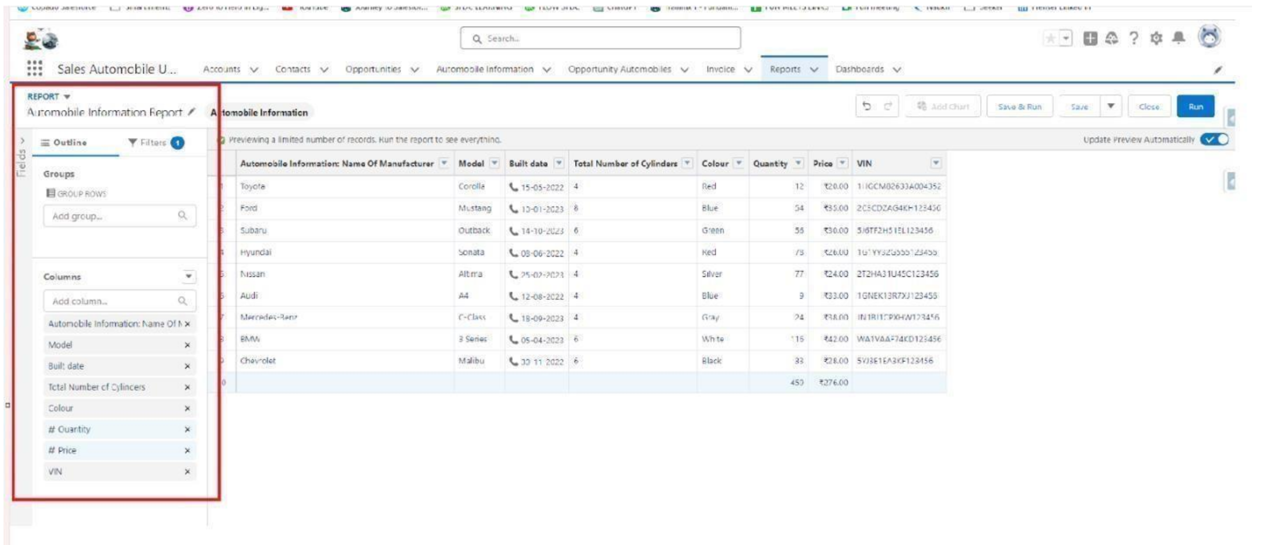
Add the Above Filter as well.

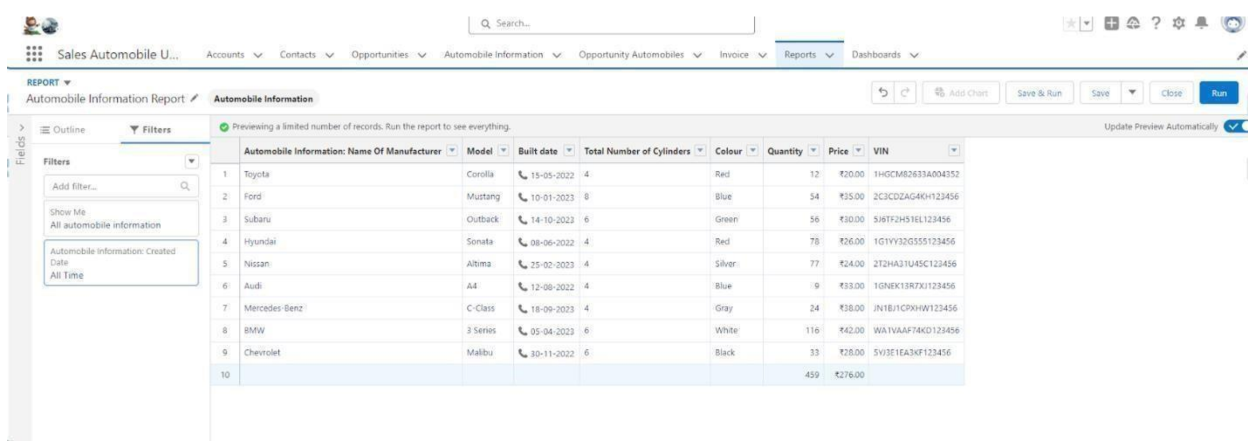
3. Save or run it.

Note: Reports may get varied from the above pictures as the data might be different.

SUBTASK 10.2: Create Report on Automobile Information

1. Create a report with a report type: “Automobile Information”.





- Create a Report by using “Opportunities with Opportunity Automobiles and Automobile” Report Type.

TASK 11- DASHBOARDS

SUBTASK 11.1: Create Dashboard

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

2. Give a Name and click on Create.

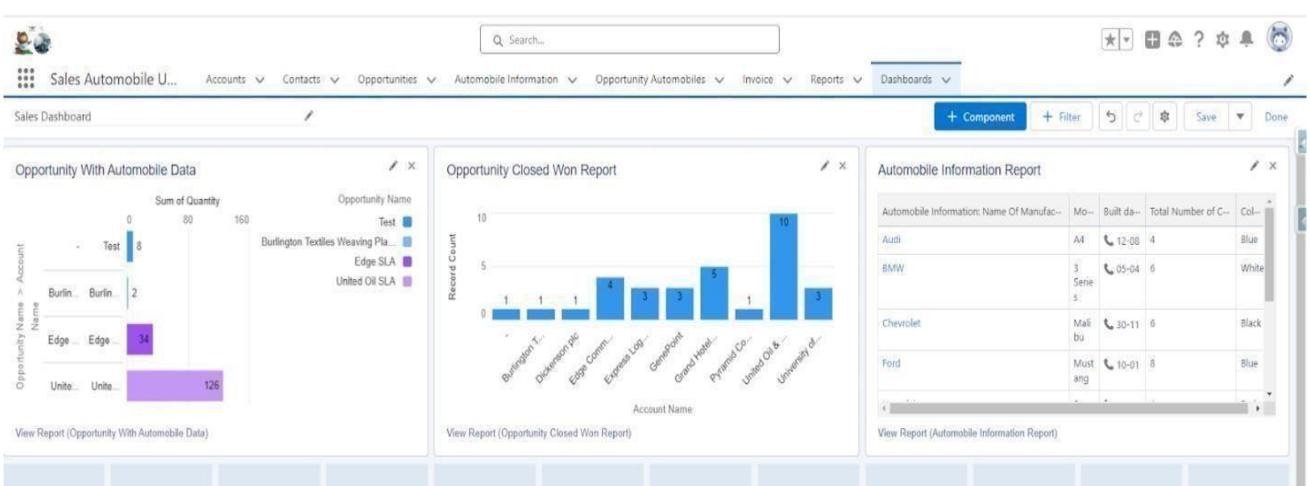
    a. Name : Automobile Sales

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:

The implementation of Salesforce CRM in the automobile sales process has proven to be a game-changer, significantly enhancing operational efficiency and customer engagement. By streamlining lead management, automating follow-ups, and integrating marketing tools, the system has improved the overall customer journey, from initial inquiry to post-sale support. The real-time inventory management feature has enabled sales teams to stay updated on vehicle availability, while the comprehensive customer profiling allows for targeted marketing strategies that boost conversion rates.

Furthermore, the powerful analytics capabilities of Salesforce provide valuable insights into sales trends and demand forecasting, supporting data-driven decision-making and strategic planning. The integration of these features has not only optimized internal processes but also fostered stronger customer relationships, leading to increased satisfaction and loyalty.

In conclusion, Salesforce CRM has not only simplified the sales cycle but also driven measurable growth for automobile sales organizations. The system's ability to unify sales, marketing, and customer service under one platform ensures long-term success, improved performance, and a competitive advantage in the automotive industry.