S.K.P ENGINEERING COLLEGE

Approved by AICTE & Affiliated to Anna University

NH 66, Somasipadi Post,Chinnakangiyanur, Tiruvannamalai,

Tamil Nadu-606611

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BACHELOR OF ENGINEERING 2024-2025

FIFTH SEMESTER

PROJECT TITLE: SALES AUTOMOBIAL USING SALESFORCE(CRM)

Project Created by: Kannan. A, Kalanithi. B, Jagan Muthu. B, Hemajothi. S

Project Reviewed by: Mr.Nasrudeen sha (AP/CSE)

College Code: 5122

Team ID :

S.K.P ENGINEERING COLLEGE

Approved by AICTE & Affiliated to Anna University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certified that this is a bonafide record of work done by,

Name : Kannan. A, Kalanithi. B, Jagan Muthu. B, Hemajothi. S

University Reg No :

Year / Semester : 3 / 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)

1.Project Overview:

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.

2. Objectives:

- **Business Goals**:
    - ❖ Increase sales conversion rates.
    - ❖ Streamline customer data management and improve accessibility.
    - ❖ Enhance communication between sales teams and customers.

- **Specific Outcomes**:
    - ❖ Implementation of a centralized CRM system.
    - ❖ Development of automated workflows for lead management and follow-ups.
    - ❖ Generation of detailed sales reports for better decision-making.

3. Salesforce Key Features and Concepts Utilized :

- **Leads and Opportunities**: Efficient tracking and management of sales pipelines.
- **Automation Tools**: Use of workflows, process builders, and Lightning Flow to automate tasks like lead assignments and follow-ups.
- **Dashboards and Reports**: Custom dashboards to visualize sales performance and identify trends.
- **Integration**: Integration with third-party tools for enhanced functionality, such as email and marketing platforms.
- **Mobile Access**: Enable sales teams to access CRM features on-the-go using Salesforce's mobile app.

4. Detailed Steps to Solution Design:

- **Data Models**:
  - Created custom objects to track automobile inventory, customer preferences, and service history.
  - Established relationships between objects for comprehensive data analysis.
- **User Interface Design**:
  - Designed user-friendly Lightning pages for sales representatives to access critical information quickly.
  - Implemented dynamic forms to customize data views based on user roles.
- **Business Logic**:
  - Developed Apex triggers for automating complex business processes.
  - Defined validation rules to ensure data integrity.

Milestone 1-Salesforce :

Introduction:

Are you new to Salesforce? Not sure exactly what it is, or how to use it? Don't know where you should start on your learning journey? If you've answered yes to any of these questions, then you're in the right place. This module is for you.

Welcome to Salesforce! Salesforce is game-changing technology, with a host of productivity-boosting features, that will help you sell smarter and faster. As you work toward your badge for this module, we'll take you through these features and answer the question, "What is Salesforce, anyway?".

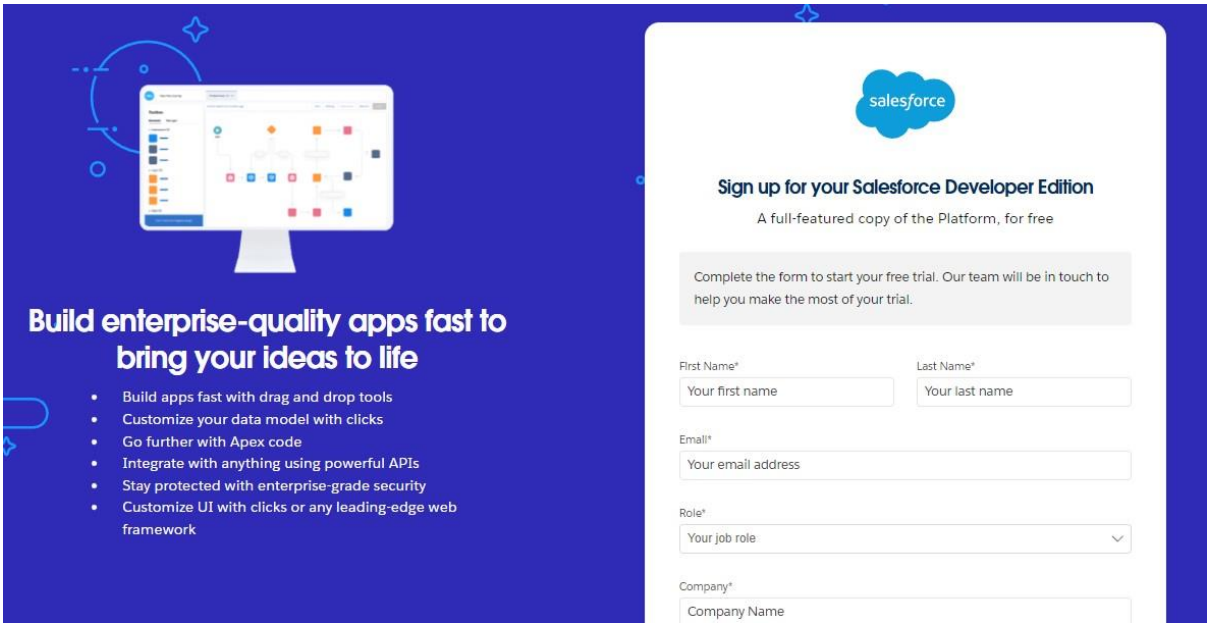
What Is Salesforce?

Salesforce is your customer success platform, designed to help you sell, service, market, analyze, and connect with your customers.Salesforce has everything you need to run your business from anywhere. Using standard products and features, you can manage relationships with prospects and customers, collaborate and engage with employees and partners, and store your data securely in the cloud.So what does that really mean? Well, before Salesforce, your contacts, emails, follow-up tasks, and prospective deals might have been organised something like this: https://youtu.be/r9EX3lGde5k

Activity 1: Creating Developer Account:

Creating a developer org in salesforce.

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



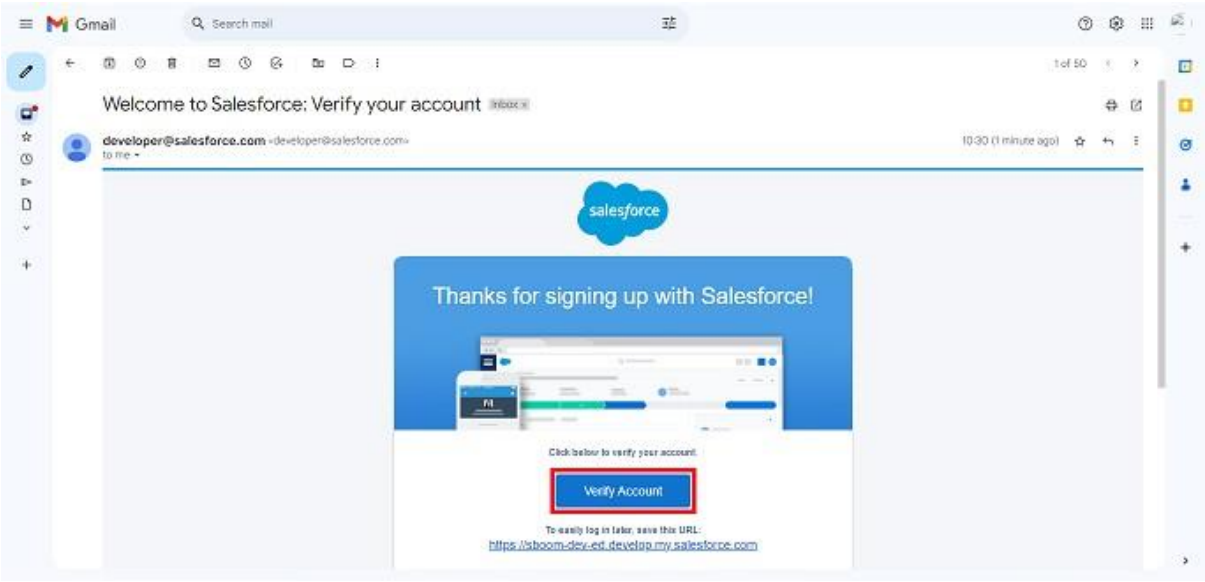
1) First name & Last name
2) Email
3) Role : Developer
4) Company : College Name
5) County : India
6) Postal Code : pin code
7) Username : should be a combination of your name and company

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.

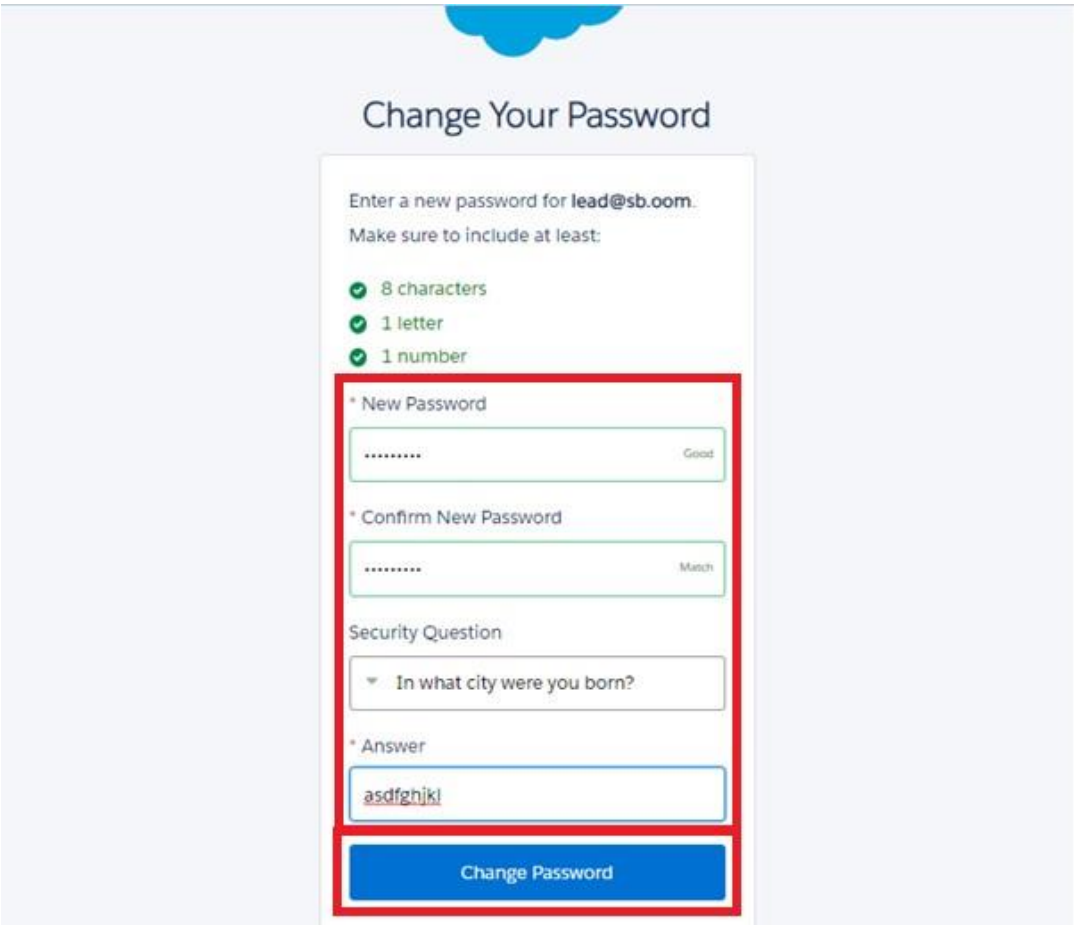
Activity 2:

Account Activation:

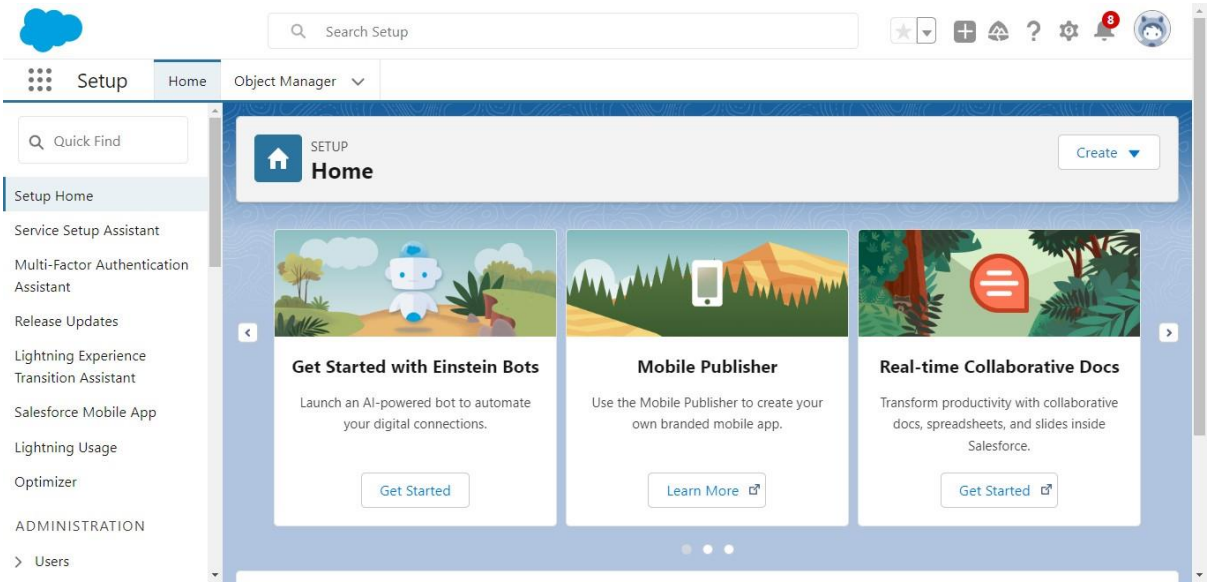
1. Go to the inbox of the email that you used while signing up. Click on the verify account to activate your account. The email may take 5-10mins.



1. Click on Verify Account
2. Give a password and answer a security question and click on change password.



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



Milestone 2- 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:

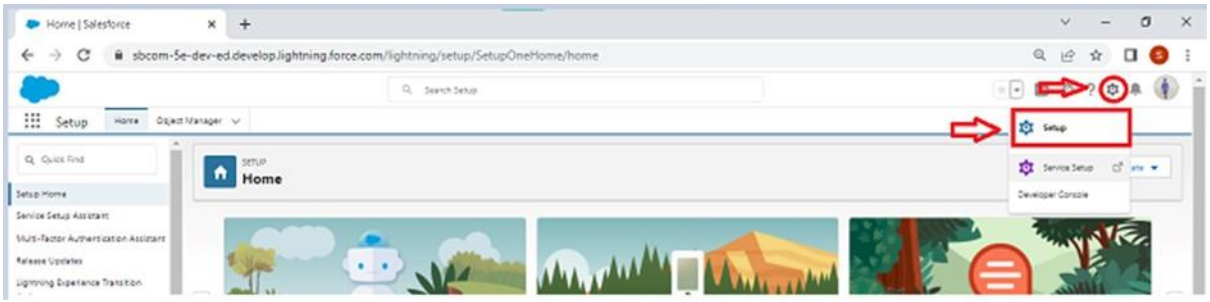
1. **Standard Objects**: Standard objects are the kind of objects that are provided by salesforce.com such as users, contracts, reports, dashboards, etc.
2. **Custom Objects:** Custom objects are those objects that are created by users. They supply information that is unique and essential to their organization.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

Use Case:

Creating an object in Salesforce organization is essential for efficient data management and process automation. By defining custom objects, businesses can structure and store data specific to their needs, enabling streamlined workflows, personalized reporting, and enhanced user experiences. Objects serve as the foundation for organizing and leveraging critical information within Salesforce.

To Navigate to Setup page:

Click on gear icon → click setup.



To create an object:

Activity 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.

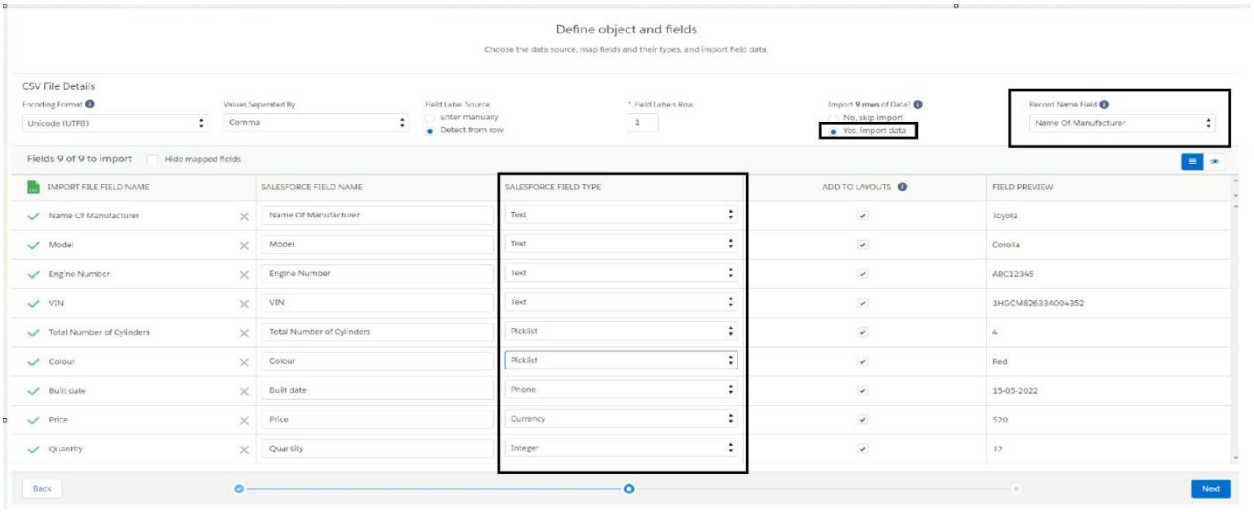
3. Click the Object Manager tab.

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.

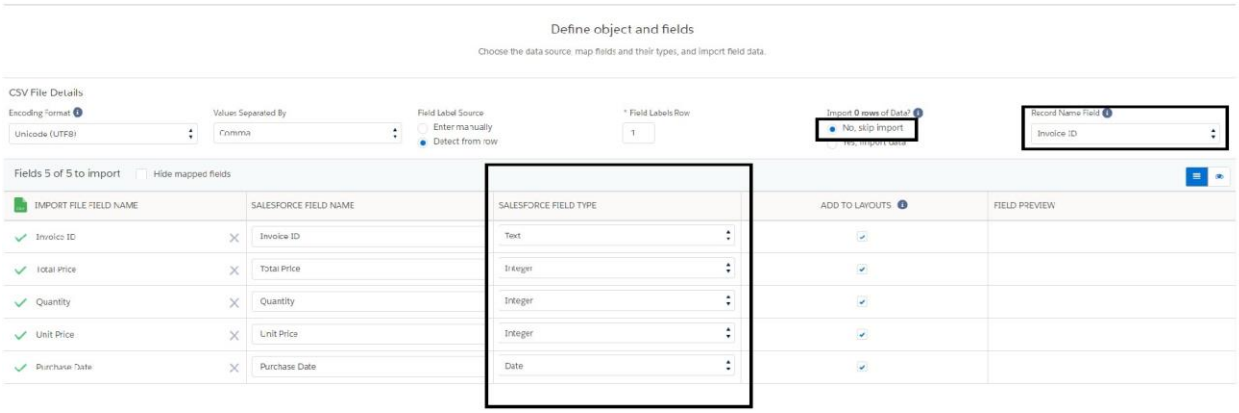


12. Click Next and enter the following settings.
13. Click Finish. The Automobile Information object is successfully created and data imported, all within minutes.

Activity 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.

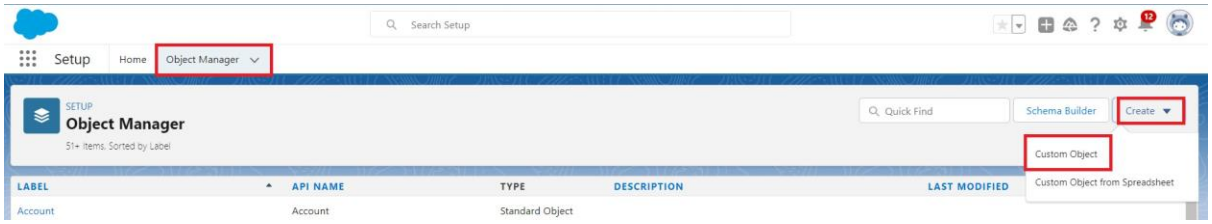


Activity 3 : Create Automobile Object

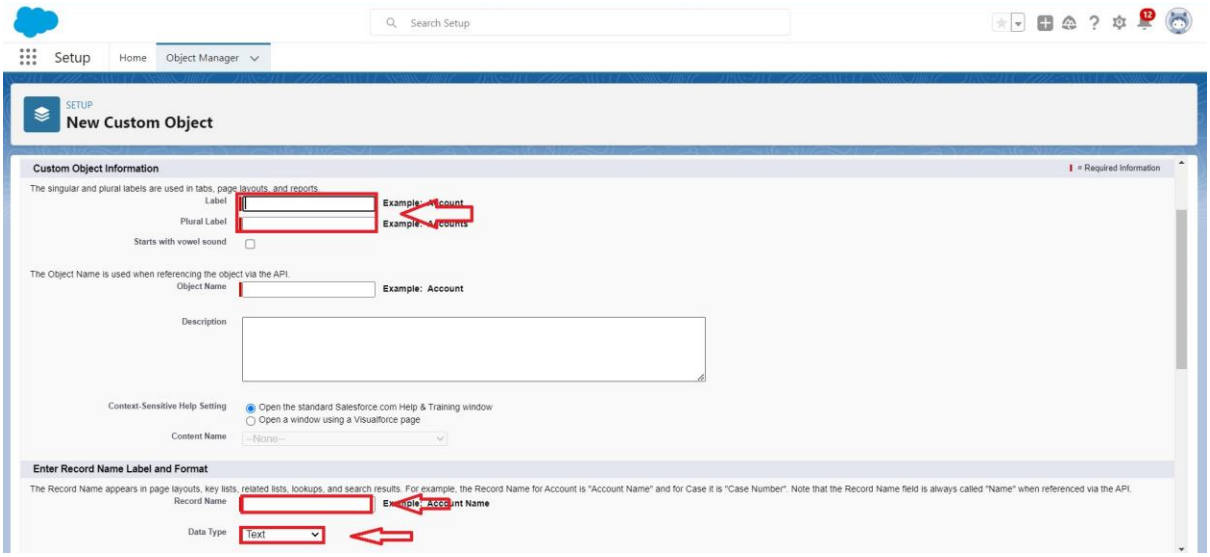
The purpose of creating an Automobile custom object is to store and manage information about Invoice.

To create an object:

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

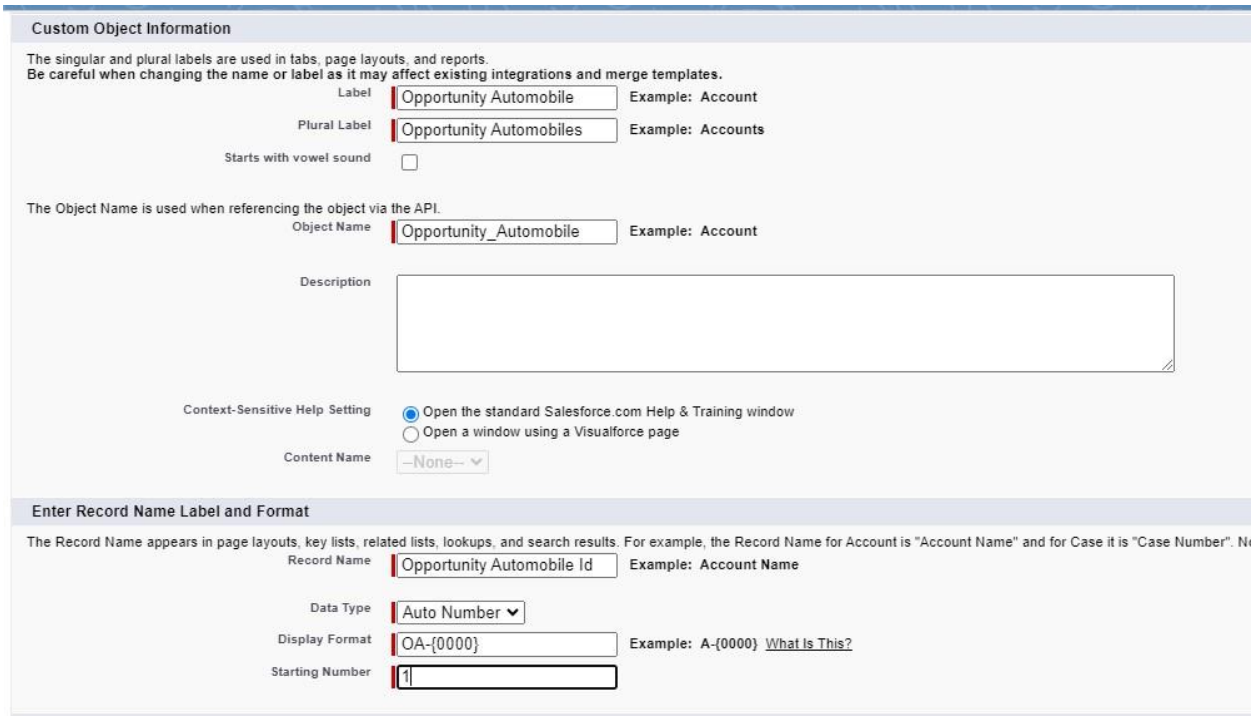


1) Enter the label name→ Opportunity Automobile
2) Plural label name→Opportunity Automobiles



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 sears 4. → **Save**

Milestone 3 - Tabs

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:

1. **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.

2. **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.

3. **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.

4. **Lightning Component Tabs**

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

5. **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.

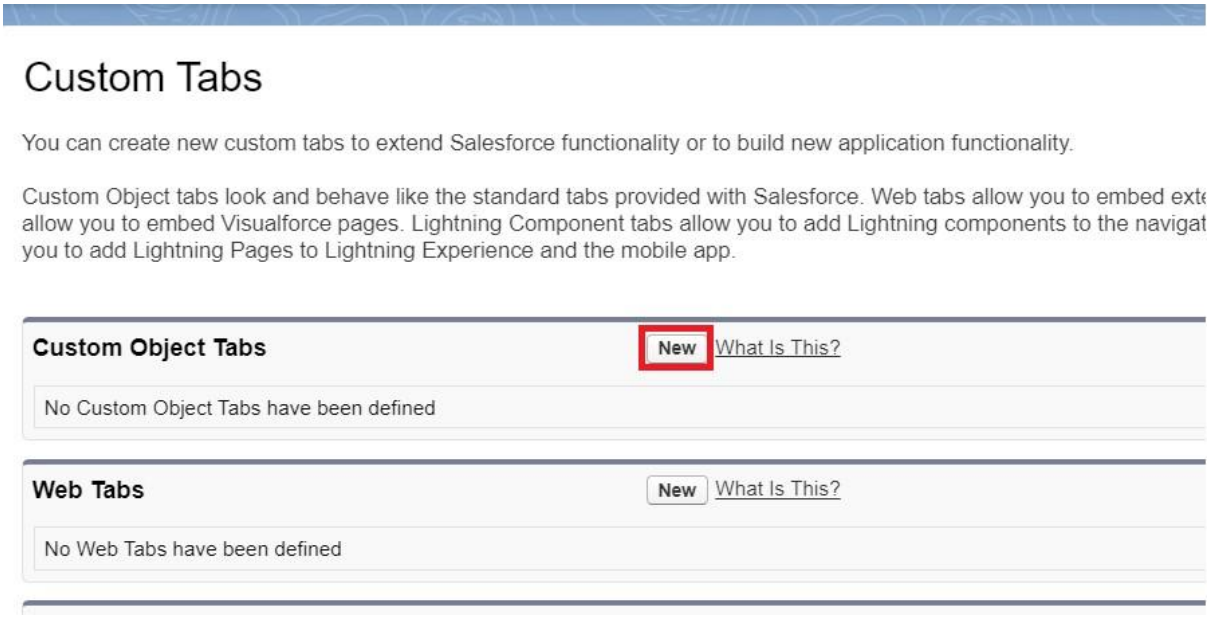
Use Case:

Creating Objects and storing organization's data is the very first step in the requirements they want. Now to access the stored data by an employee from the organization Admin needs to create Tabs. By designing a dedicated Tab, businesses can improve user experience, simplify navigation, and provide quick access to critical information, enhancing productivity and ensuring efficient utilization of Salesforce's capabilities.

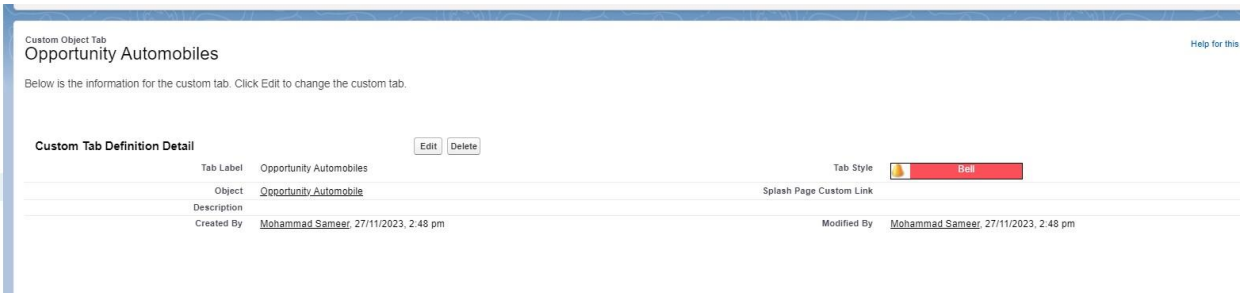
Activity 1: Creating a Custom Tab

To create a Tab: (Opportunity Automobile)

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.

Milestone 4- The Lightning App:

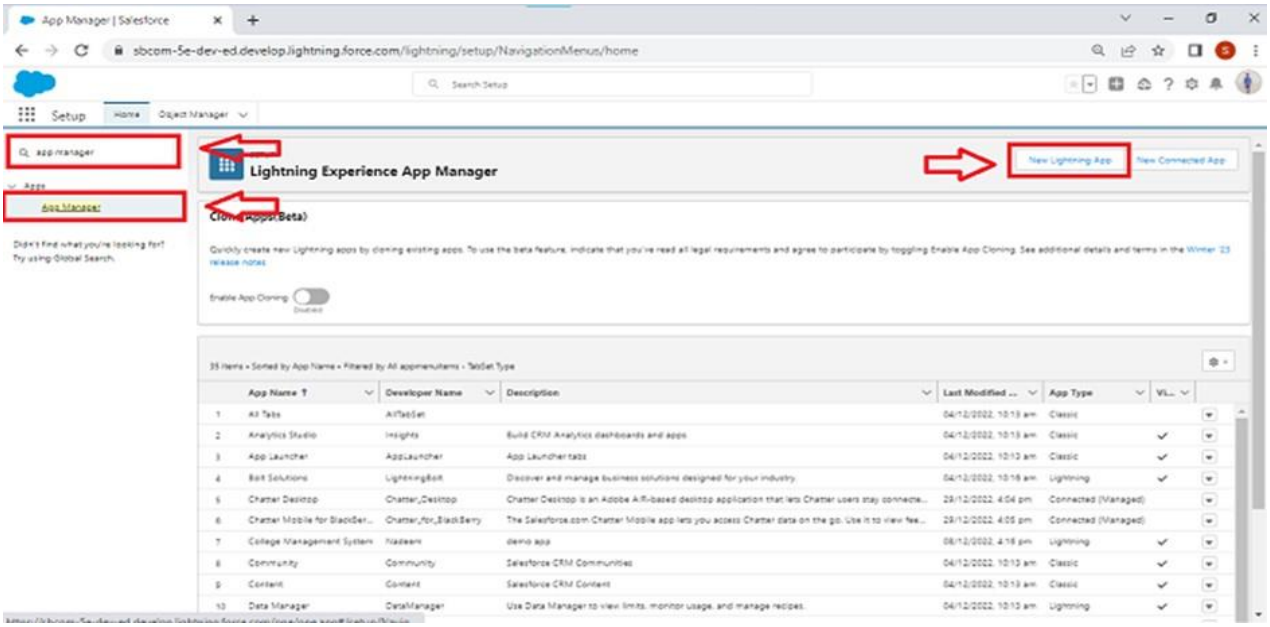
An app is a collection of items that work together to serve a particular function. In Lightning Experience, Lightning apps gives users access to sets of objects, tabs, and other items all in one convenient bundle in the navigation bar. Lightning apps let you brand your apps with a custom color and logo. 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.

Use Case:

Well done you have reached close to your organizational requirement by creating the objects to store the organization's data. Making a database for an organization is just not enough to reach out the requirements, the task is how the users at the organization can access the objects you have created for them. As an Admin for the organization it's your duty to make sure every user of the organization is able to access the data modeling structure.

Activity 1: Create a Lightning App

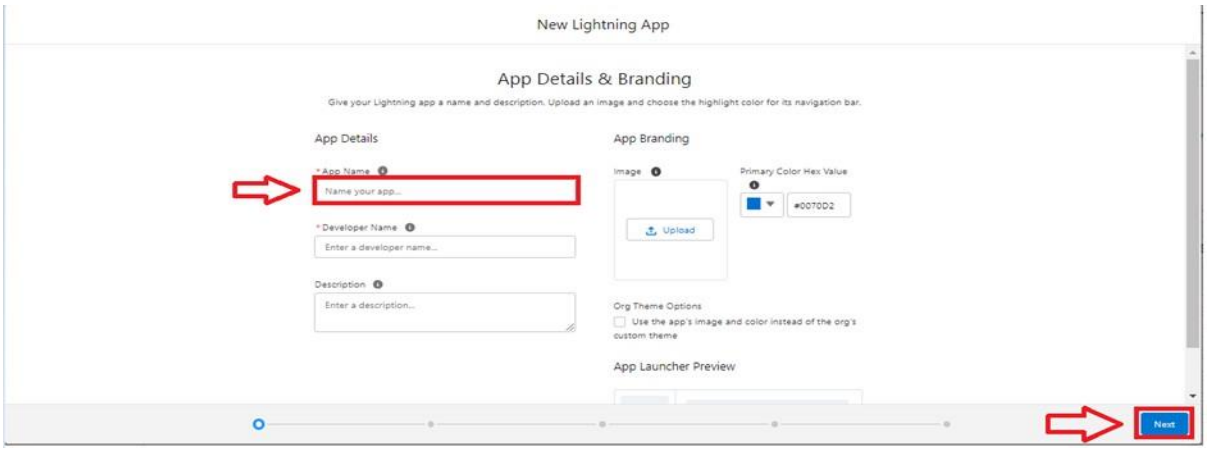
1. Go to setup page → search "app manager" in quick find → select "app manager" → click on New lightning App.



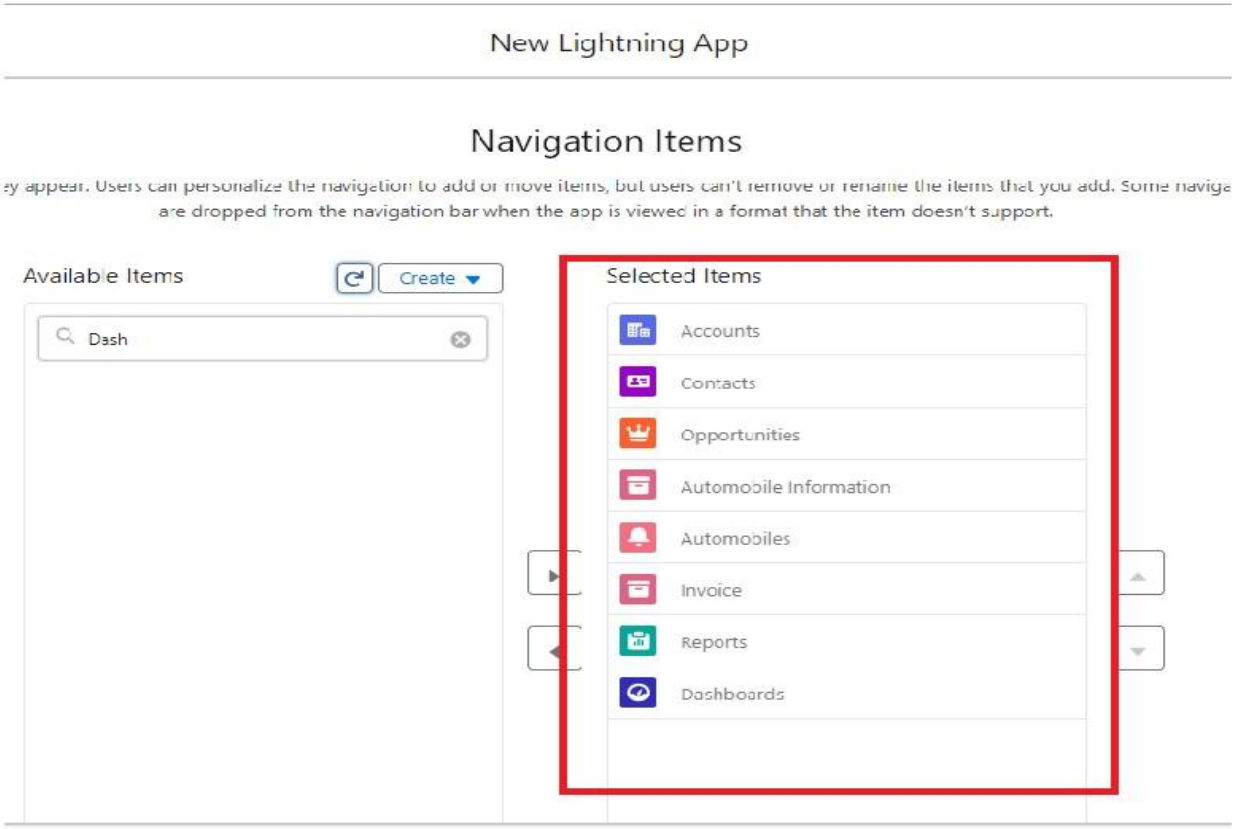
2. Fill the app name in app details and branding as follow

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

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

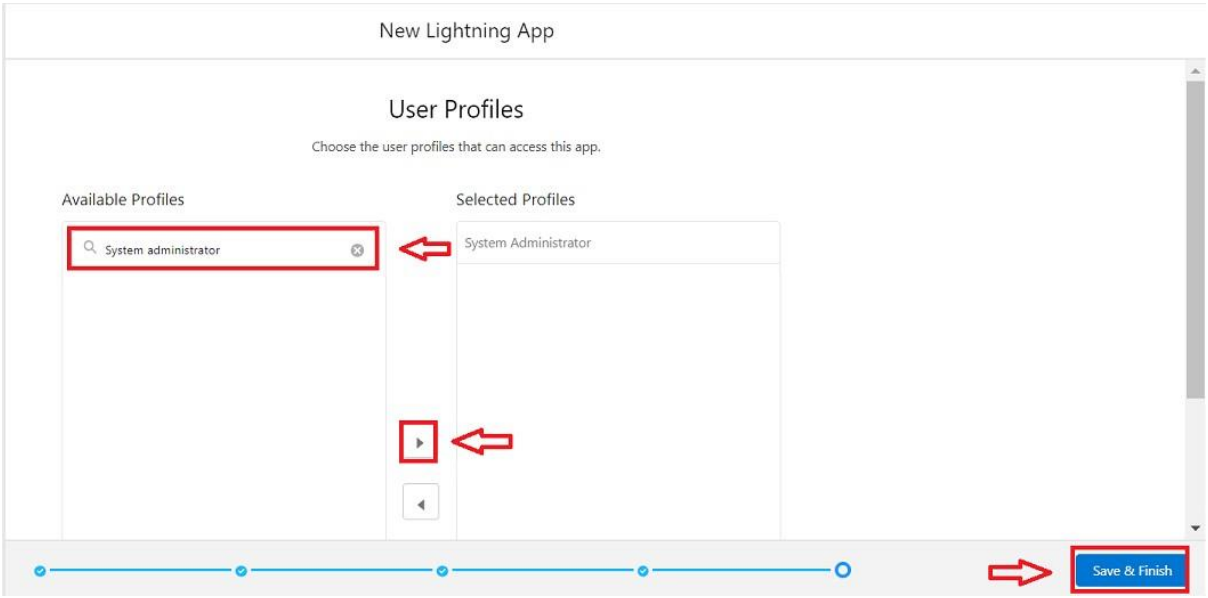


4. To Add Navigation Items:



5. 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.



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

Milestone 5: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

Standard 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.

Use Case:

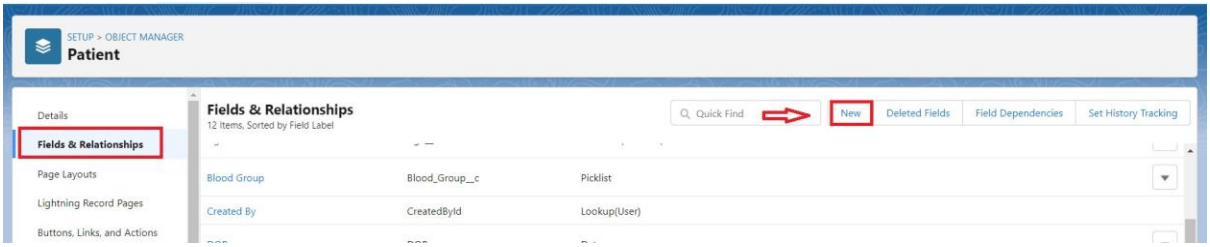
Now it's time for you to think out of the box for your organization. You have successfully created the database objects for the organization but now all eyes turn on you as you have to define what sort of information the objects store which you have created. As a life saver of your organization you come up with the idea of creating fields to store different types of data.

Activity 1: Creating Opportunity Master Detail Relationship Field in Opportunity AutoMobile Object

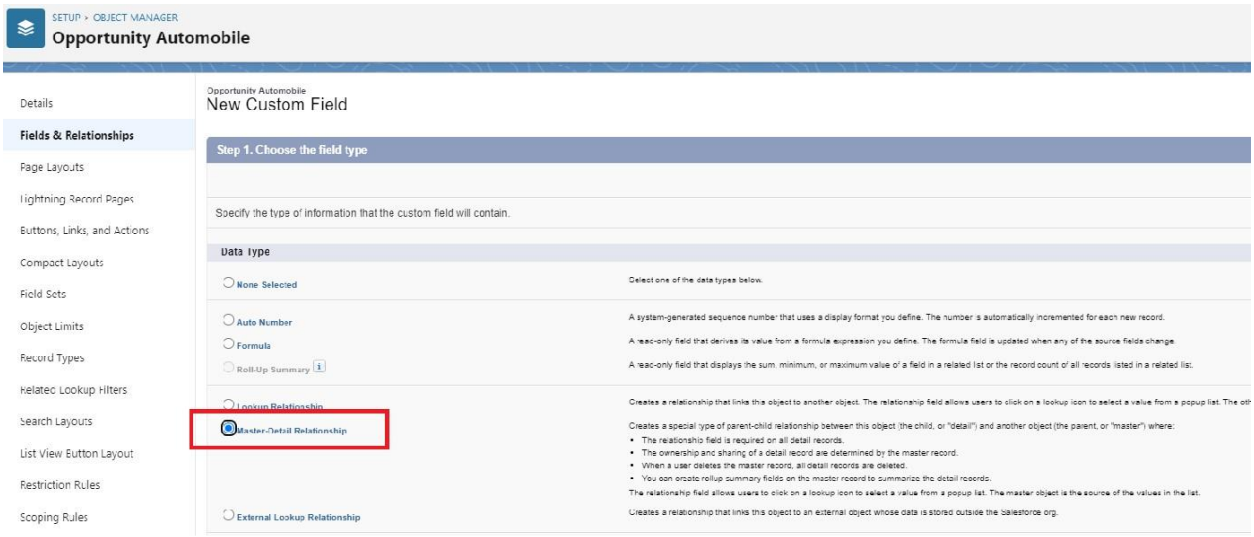
To create fields in an object:

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

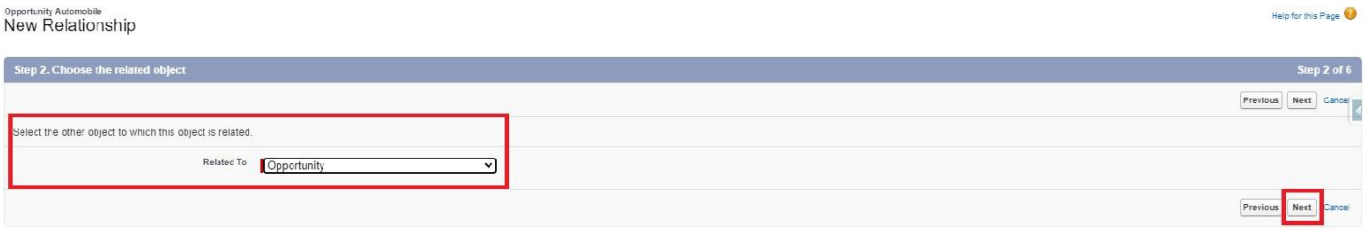
2. Now click on “Fields & Relationships” → New



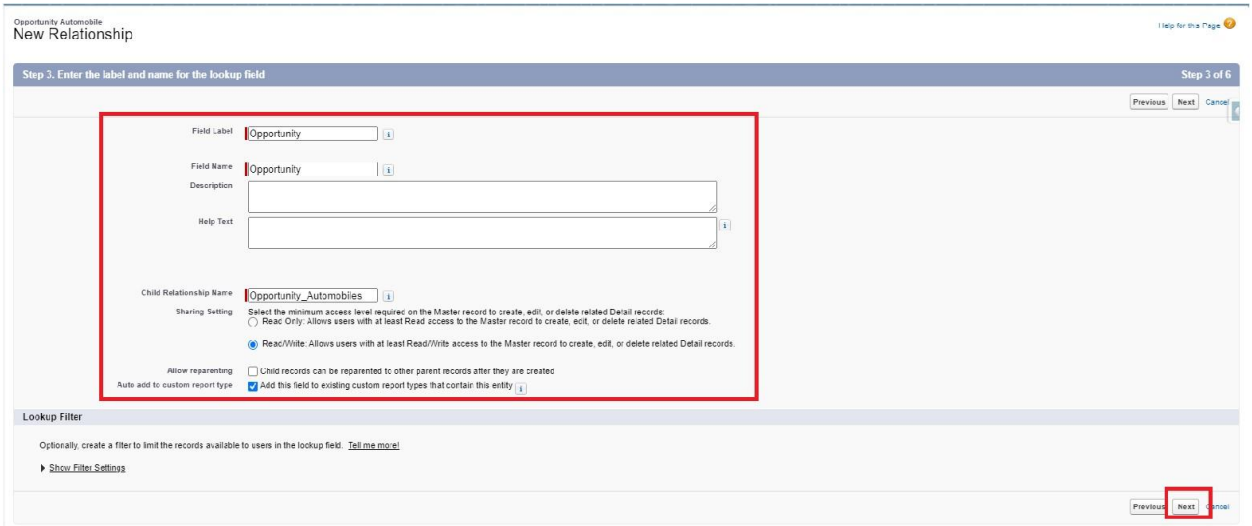
3. Select Data type as “Master Details Relationship”.



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.

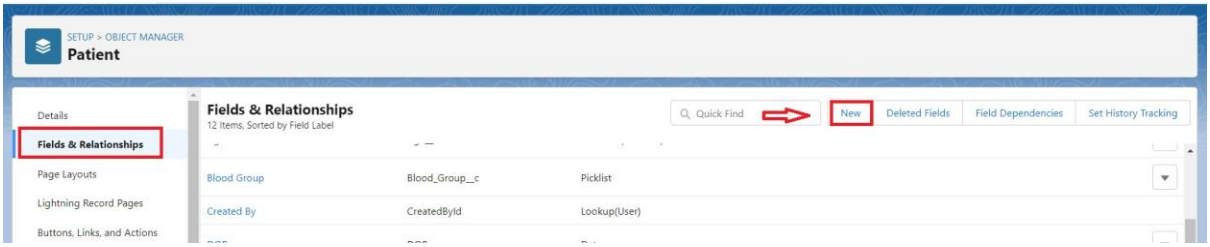


Activity2: Creating the AutoMobile Information Lookup Fieldin Opportunity Automobile Object

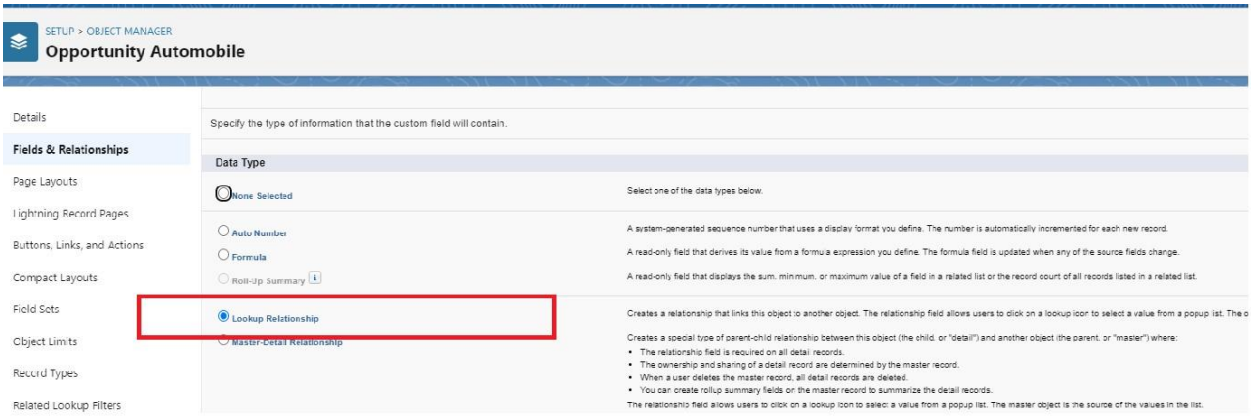
To create fields in an object:

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

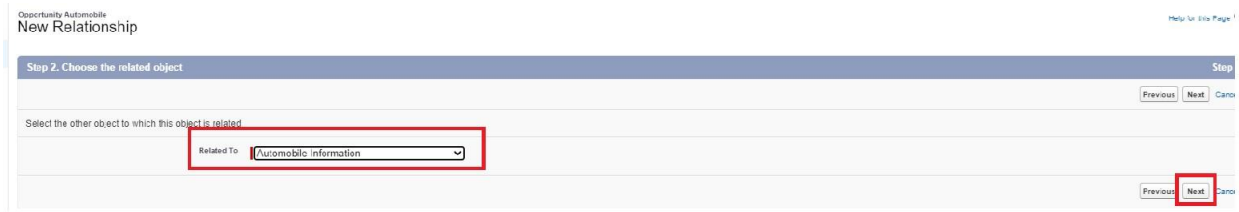
2. Now click on “Fields & Relationships” → New



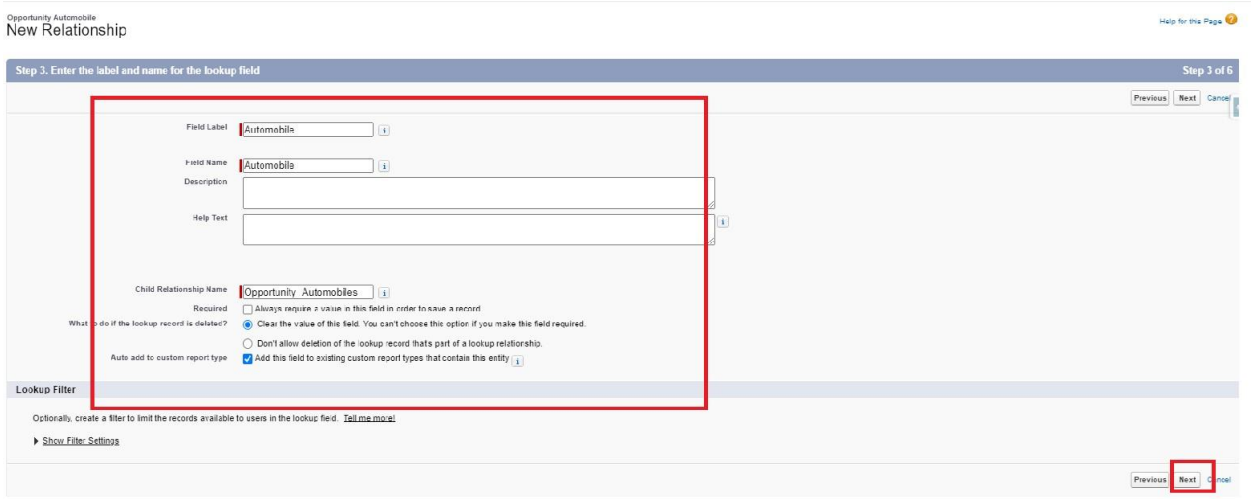
3. Select Data type as “Lookup RelationShip”.



4. Click on Next



5. Fill the above as following:
   a. Field Label: Automobile
   b. Field Name : Automobile
6. Click on Next → Next → Save and new.

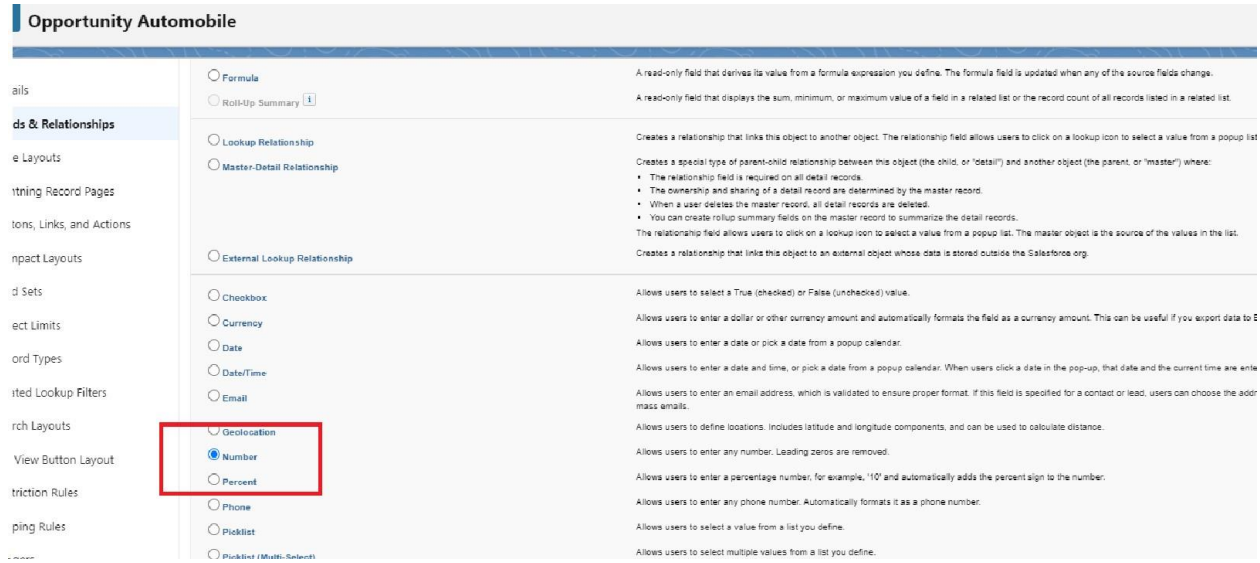


Activity 3: Creating Quantity Number Field in Opportunity Automobile Object

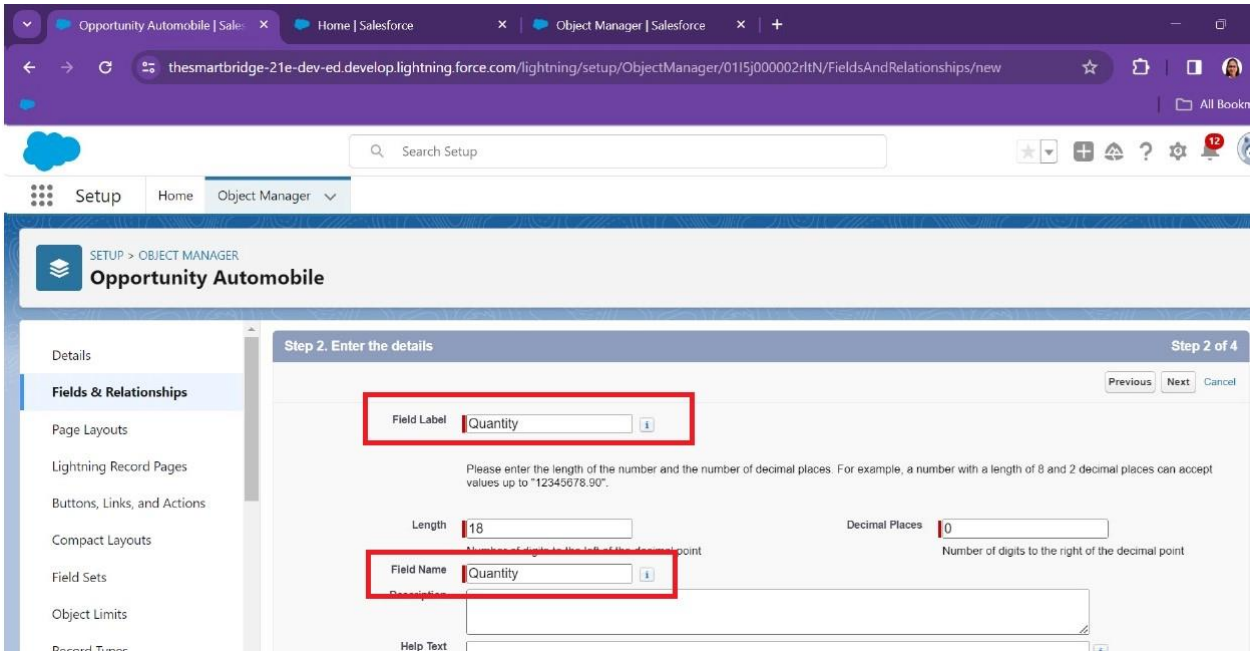
To create fields in an object:

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



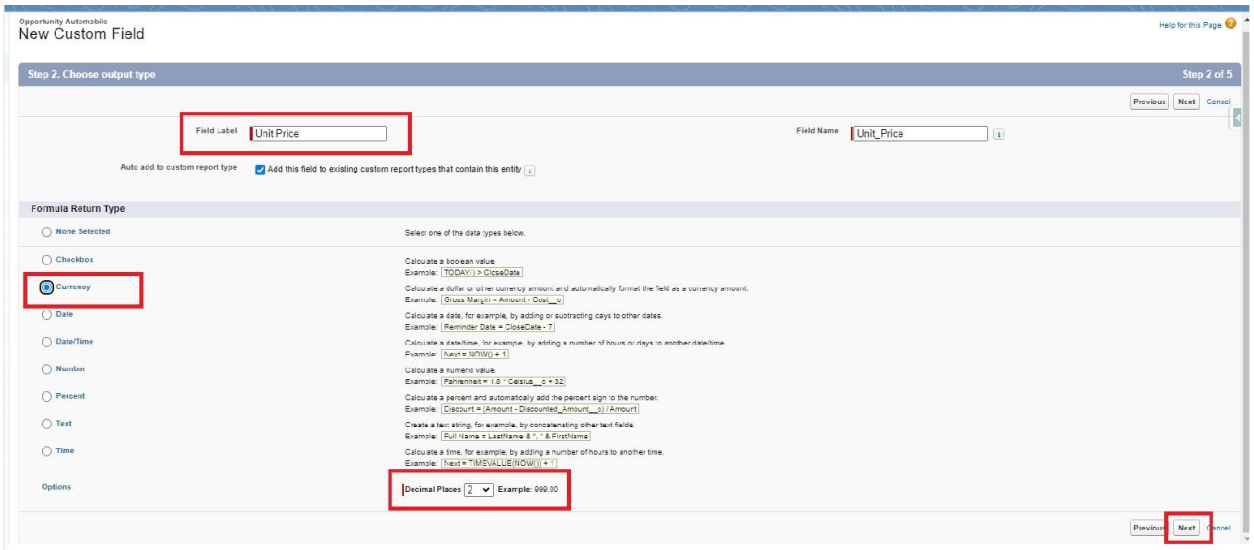
4. Check that Required Check box.
5. Click Next → Next→ Save & New.

Activity 4 : Creating Formula Field in Opportunity Automobile Object

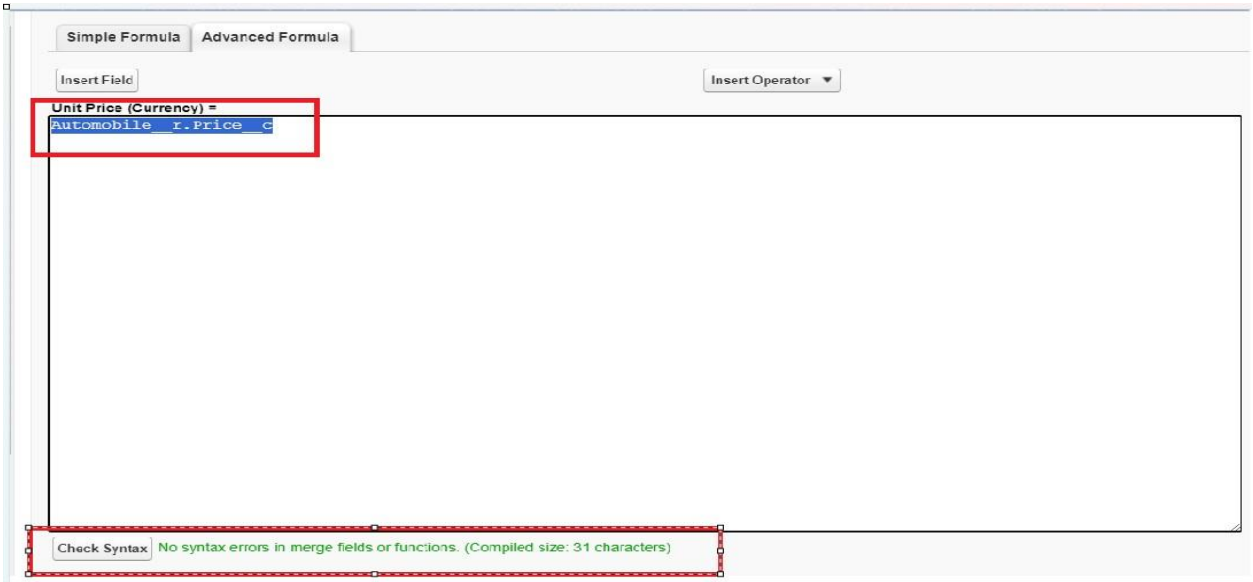
To create fields in an object:

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**

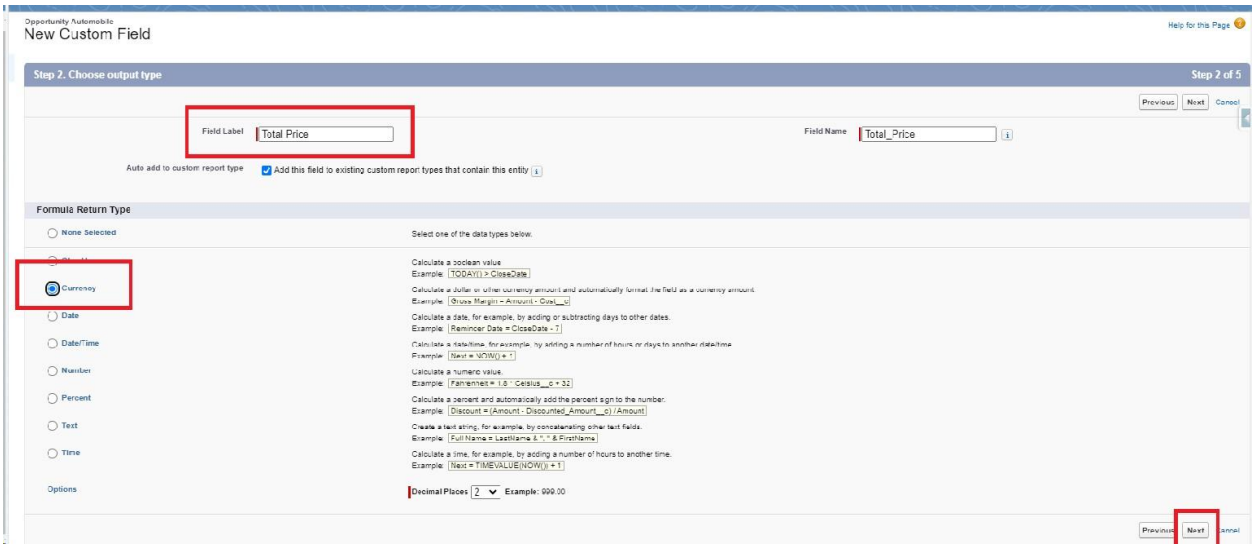


5. click “Check Syntax” and Next → Next→ Save & New.

Activity 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.
2. Now click on “Fields & Relationships” → New.
3. Select Data type as “Formula” and click Next.
4. 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.



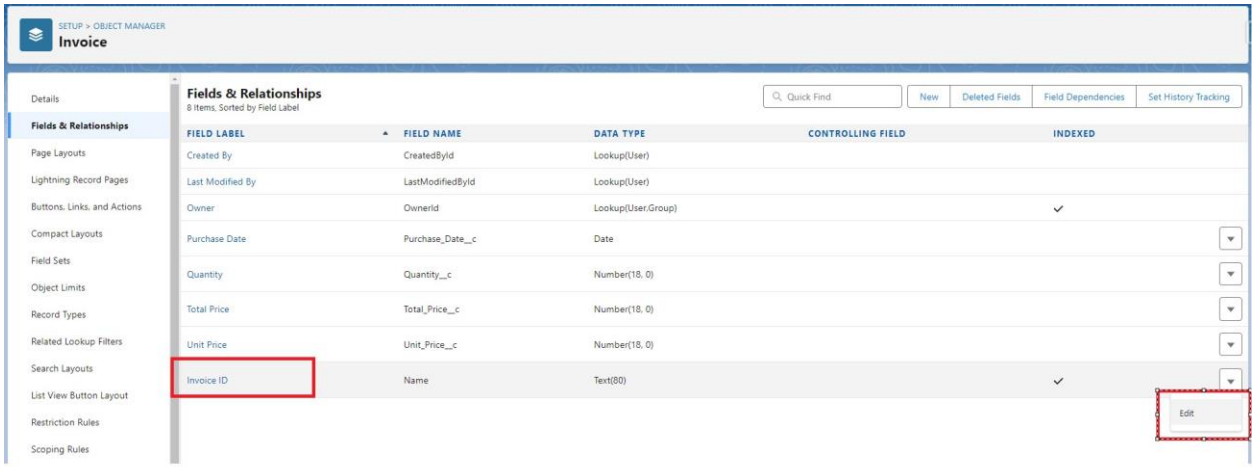
5.Under Advanced Formula write down the formula : **Unit_Price__c * Quantity__c**

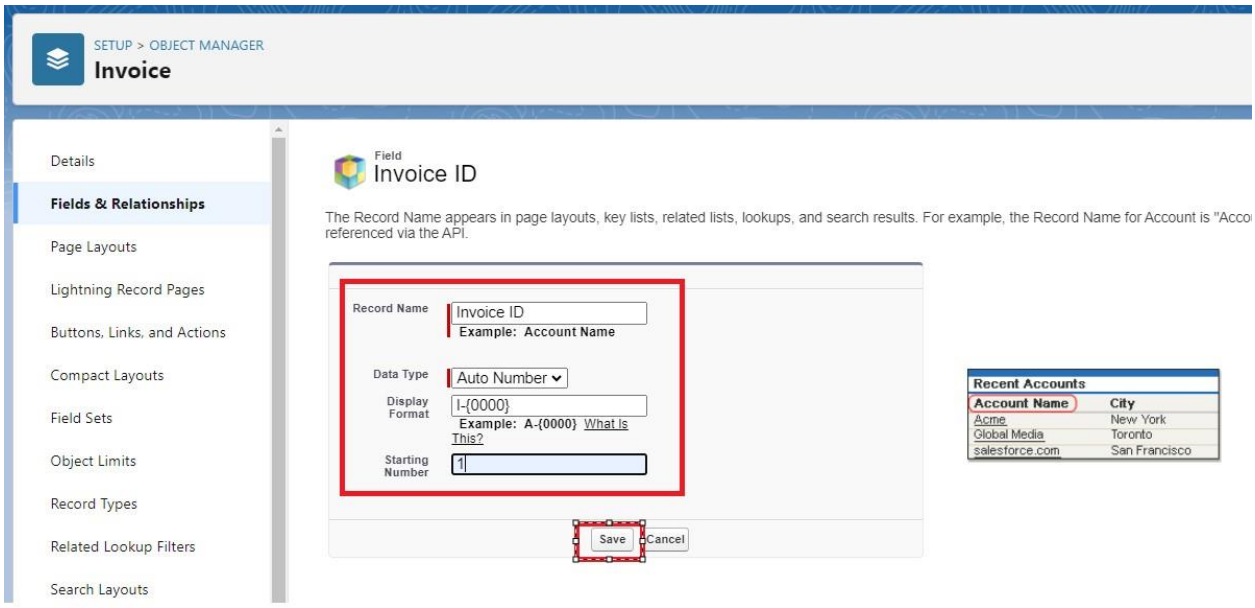
6. click “Check Syntax” and Next → Next→ Save.

Activity 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:-
4. Click Save.

Activity 7 : Creating Remaining Fields in Objects

Now create the remaining fields using the data types mentioned.

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

Milestone 6 : 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.

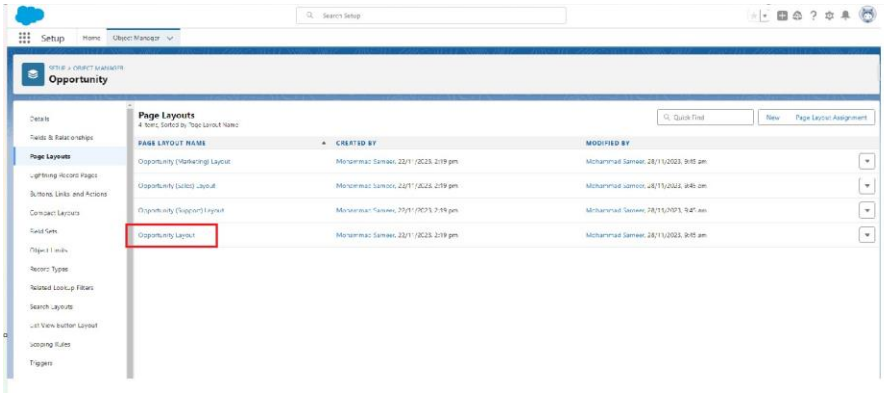
Use Case:

Hurray!! you have completed the data model structure for your organization but while looking at the detailed and edit pages it seems to be so clumsy, so decide to organize the page in a pleasant way for the sake of good and pleasant appearance and assemble all different kinds of information in different sections in order.

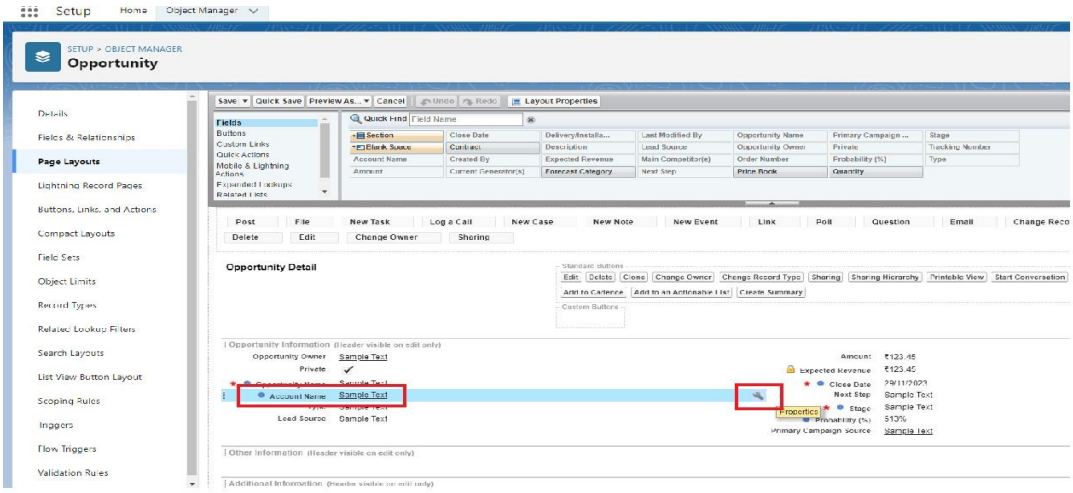
Activity 1: Edit the Page layout for Opportunity Object

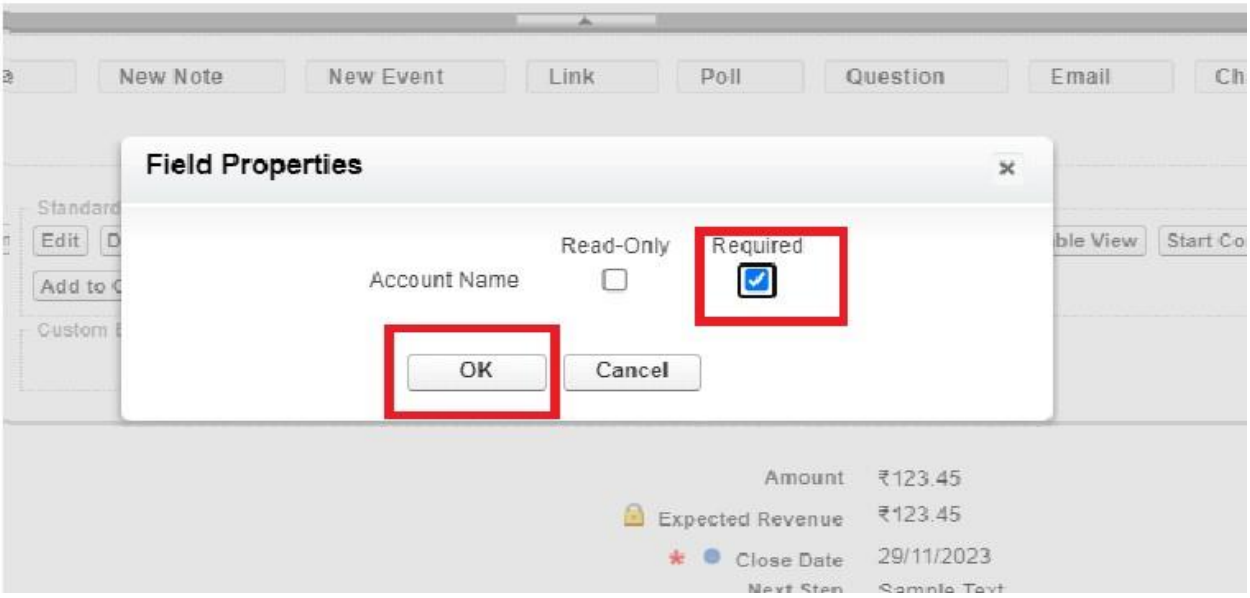
Step 1: Go to Setup → Click on Object Manager →On the search bar, select Opportunity Layout. You can notice Page Layouts on the left panel

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.





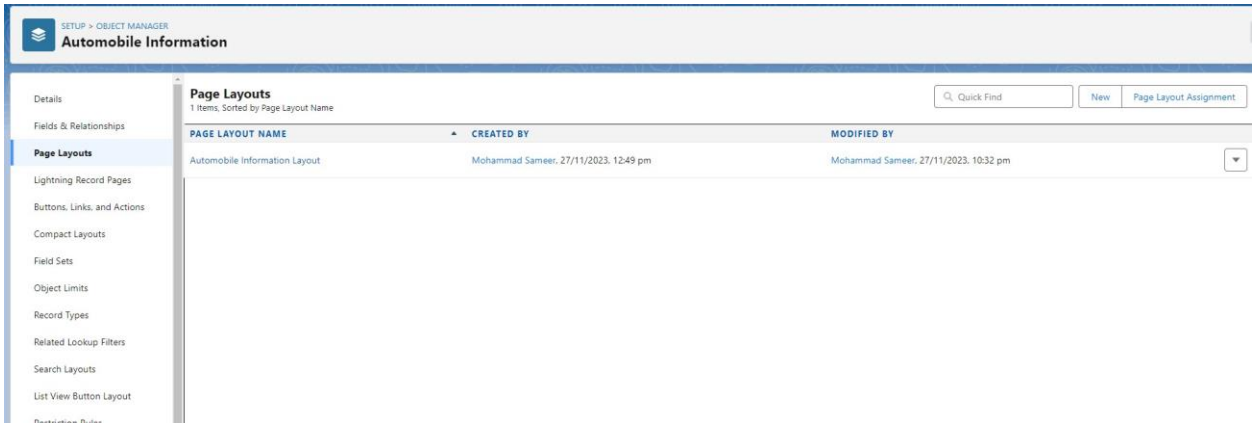
Step 4: check the Required box for Account name and click on Ok.

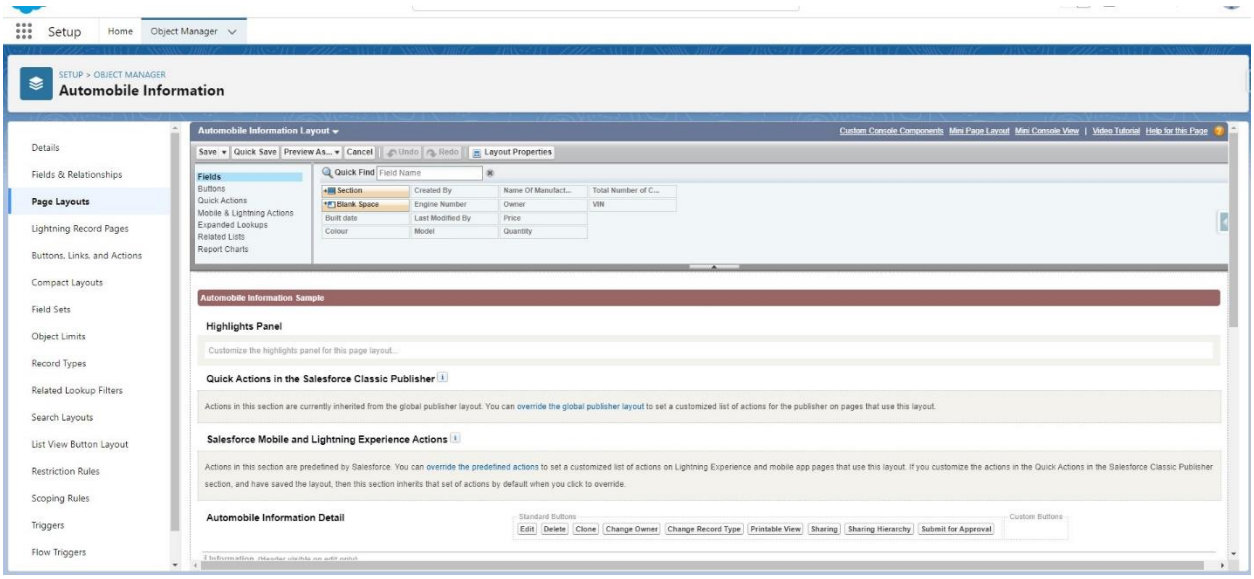
Step 5: Click on Save.

Activity 2: Edit the Page layout for Automobiles Information

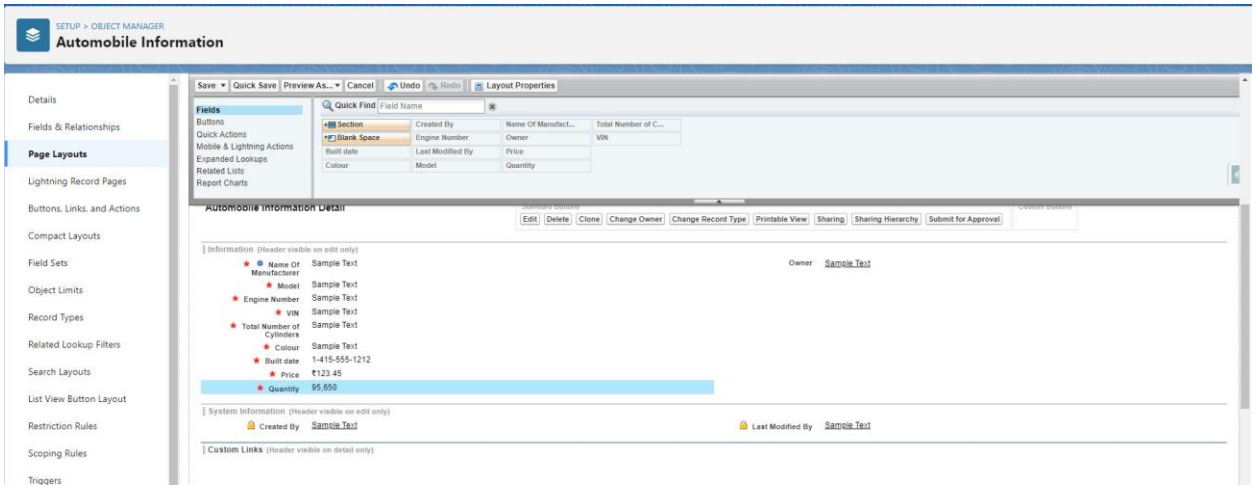
Step 1: Go to Setup → Click on Object Manager →On the search bar, select Automobile Information. You can notice Page Layouts on the left panel

Step 2: Click on Page Layouts. Click on ‘Automobile Information Layout’.

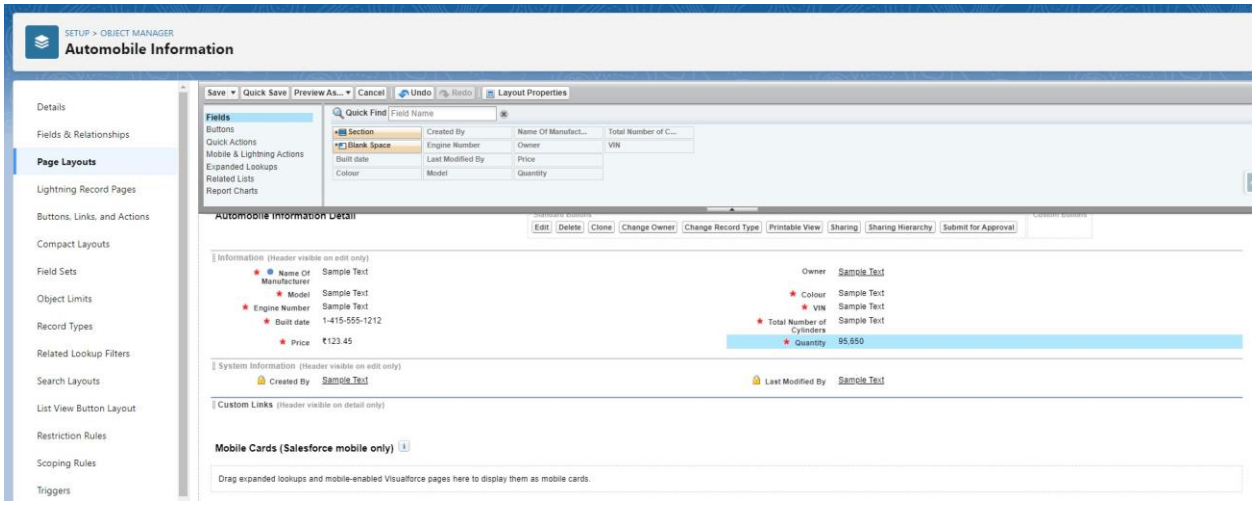




Step 3: Just Go for each one field of Automobile Information Object, Click on Gear Icon and mark as Required just as Done for Above Account Object. After required is done it will show the red color as given in below image.



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



Step 5 : Click on Save.

Milestone 7: 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:

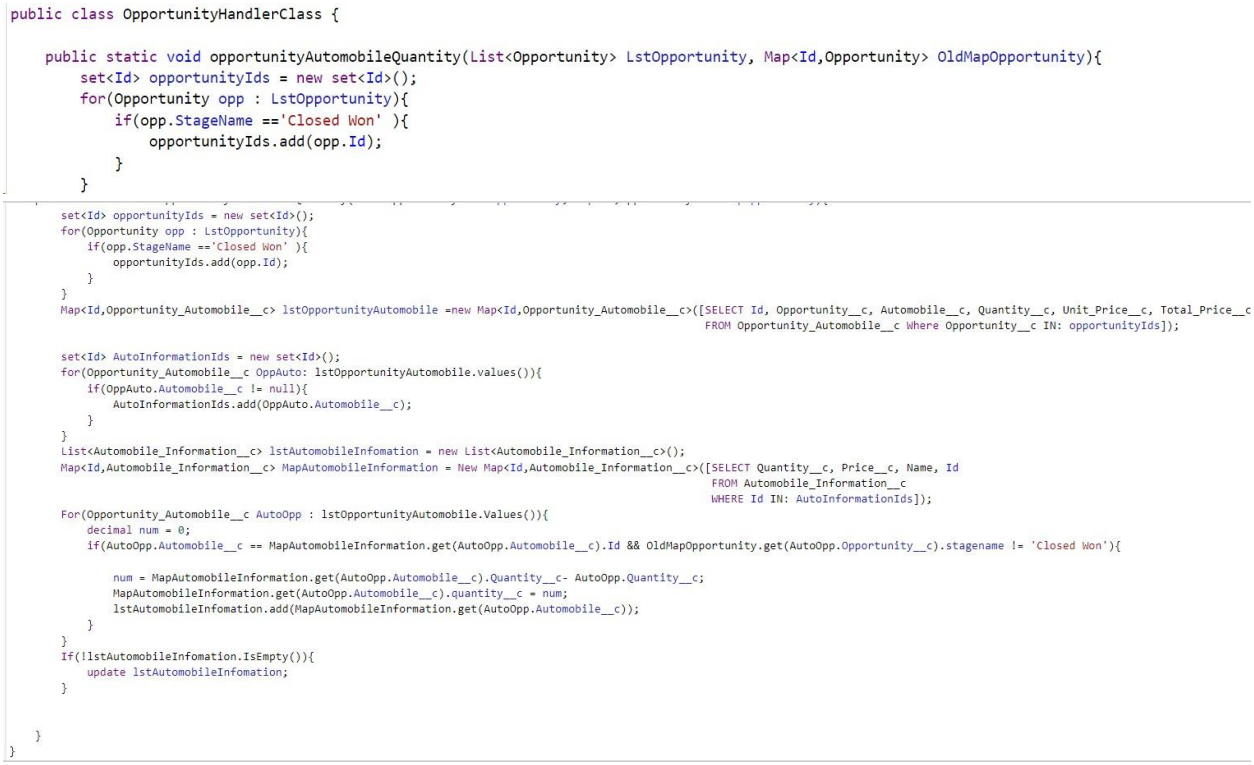
Before Trigger: This type of trigger in Salesforce is used either to update or validate the values of a record before they can be saved into the database. So, basically, the before trigger validates the record first and then saves it. Some criteria or code can be set to check data before it gets ready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

Activity- 1:

UseCase : 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 “OpportunityHandlerClass ”.



```
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__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;
        }


    }
}
```

Code:

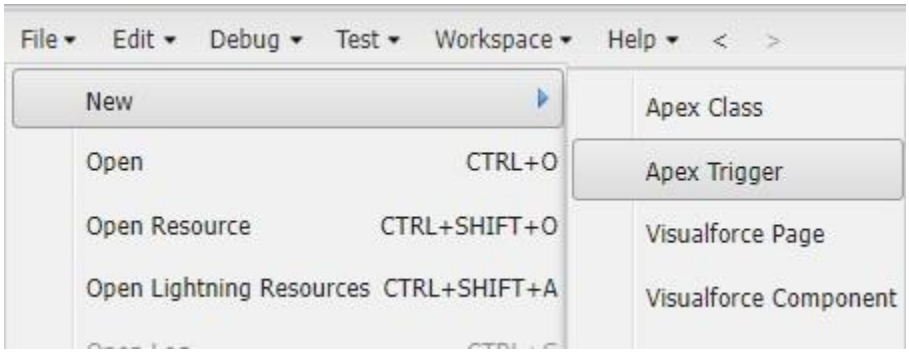
```
public class OpportunityHandlerClass {

   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 :

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



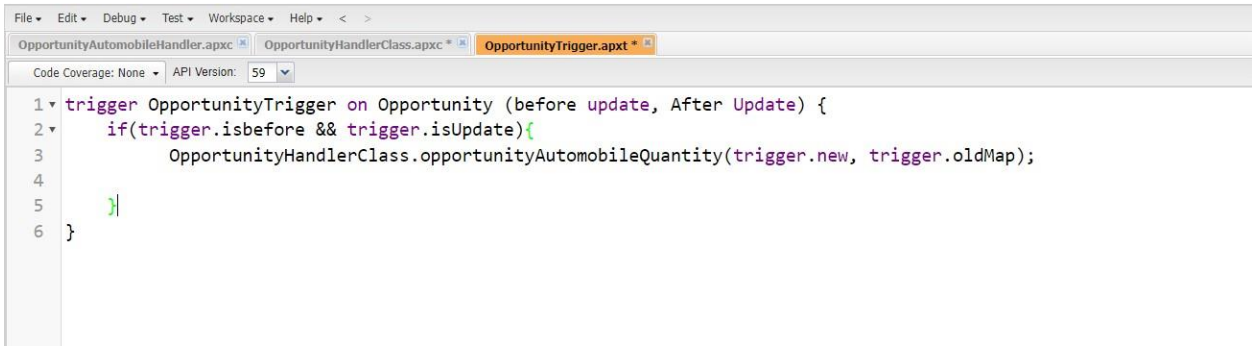
Syntax For creating trigger :

The syntax for creating trigger is :

Trigger [trigger name] on [object name](Before/After event){
}

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.



Code: trigger OpportunityTrigger on Opportunity (before update, After Update) { if(trigger.isbefore && trigger.isUpdate){

OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);

}

}

Activity- 2:

UseCase : 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 “OpportunityAutomobileHandler ”.

Code:

public class OpportunityAutomobileHandler {

public static void quantityErrorOnAutomobileInformation(List<Opportunity_Automobile__c> lstOpportunityAutomobile){ set<Id>

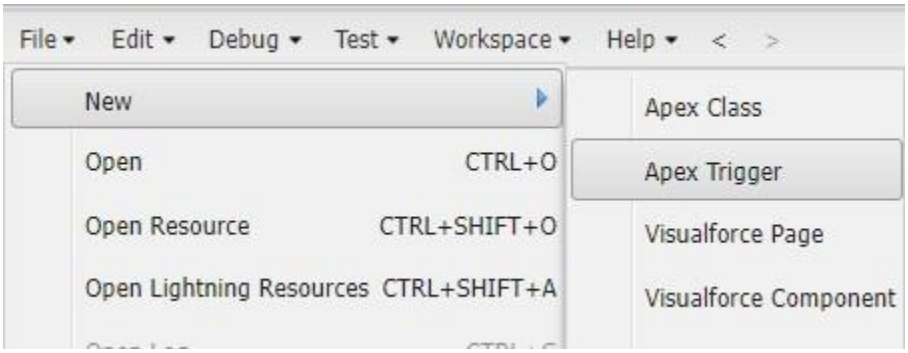
AutomobileIds = new Set<Id>();

```
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 :

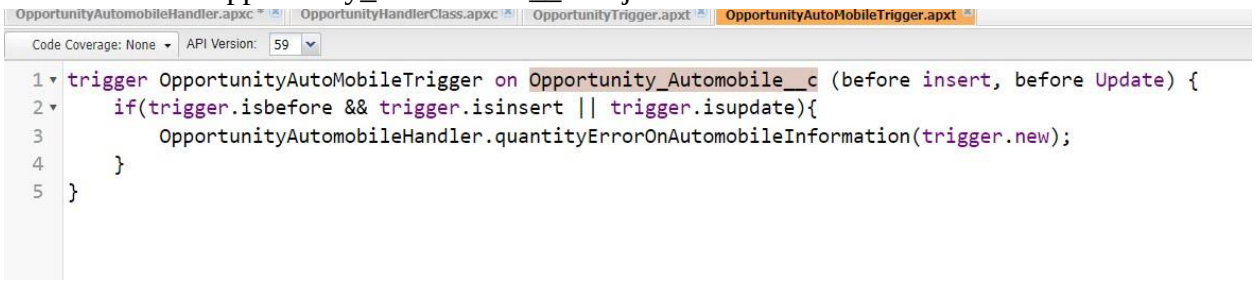
How to create a new trigger :

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 : OpportunityAutoMobileTrigger
6. sObject : Opportunity_Automobile__c



Trigger :

Handler for the Opportunity_Automobile__c Object



Code: trigger OpportunityAutoMobileTrigger on Opportunity_Automobile__c (before insert, before

Update) {

```
  if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
    OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(trigger.new);
  }
}
```

Activity- 3 :

UseCase : 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”.

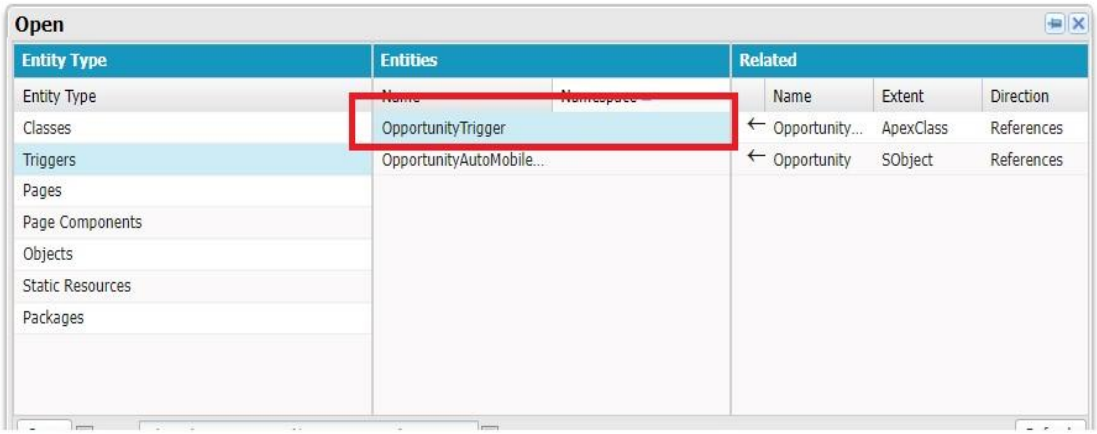
```
public class InvoiceCreation {
    public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity> lstOpportunity, Map<Id,Opportunity>OldMapOpportunity){
        set<Id> oppIds = new Set<Id>();
        For(Opportunity opp : lstOpportunity){
            if(Opp.StageName == 'Closed Won' && 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;
        }
    }
}
```

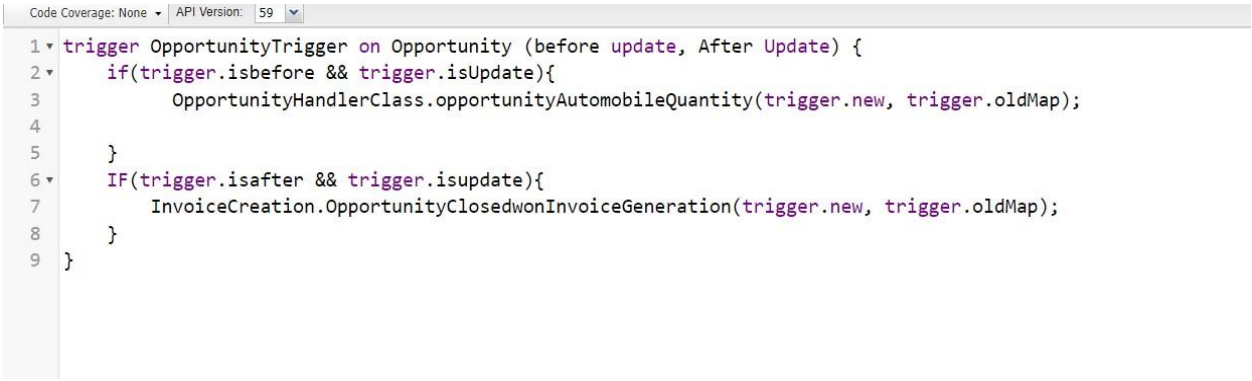
```
public class InvoiceCreation {
   public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity>
lstOpportunity, Map<Id,Opportunity>OldMapOpportunity){
      set<Id> oppIds = new Set<Id>();
      For(Opportunity opp : lstOpportunity){ if(Opp.StageName == 'Closed Won' &&
         OldMapOpportunity.get(opp.Id).StageName !=
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 OpportunityTrigger.





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);
   }
}
```

Activity- 4 :

UseCase : 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.');
                }
            }
        }
    }
}
```

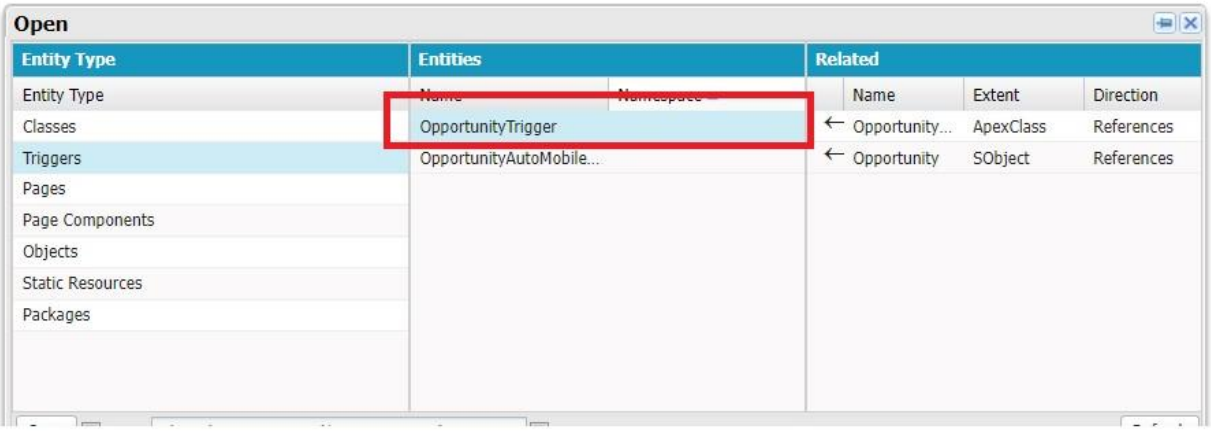
Trigger:

```
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”.

1) Go on files and click on open.
2) Click on triggers.
3) Double click on OpportunityTrigger.



Trigger Code :

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);
    }
}
```

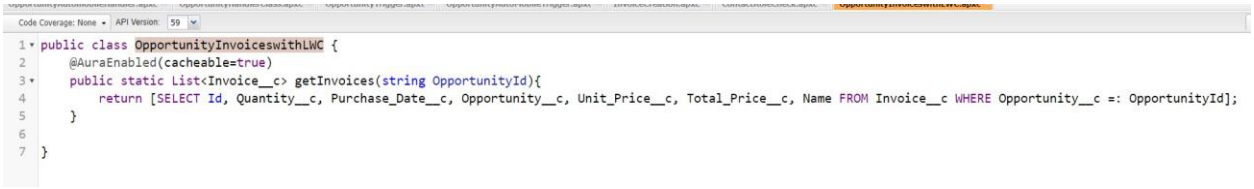
Trigger:

```
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);
   }
}
```

Milestone 8: LWC Component:

Activity- 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.
4. In the toolbar, you can see FILE.
5. Click on it and navigate to new and create New apex class.
6. Name the class as “OpportunityInvoiceswithLWC ”.



Code:

```
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];
   }

}
```

Activity- 1: 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
```

Activity- 2 : 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.

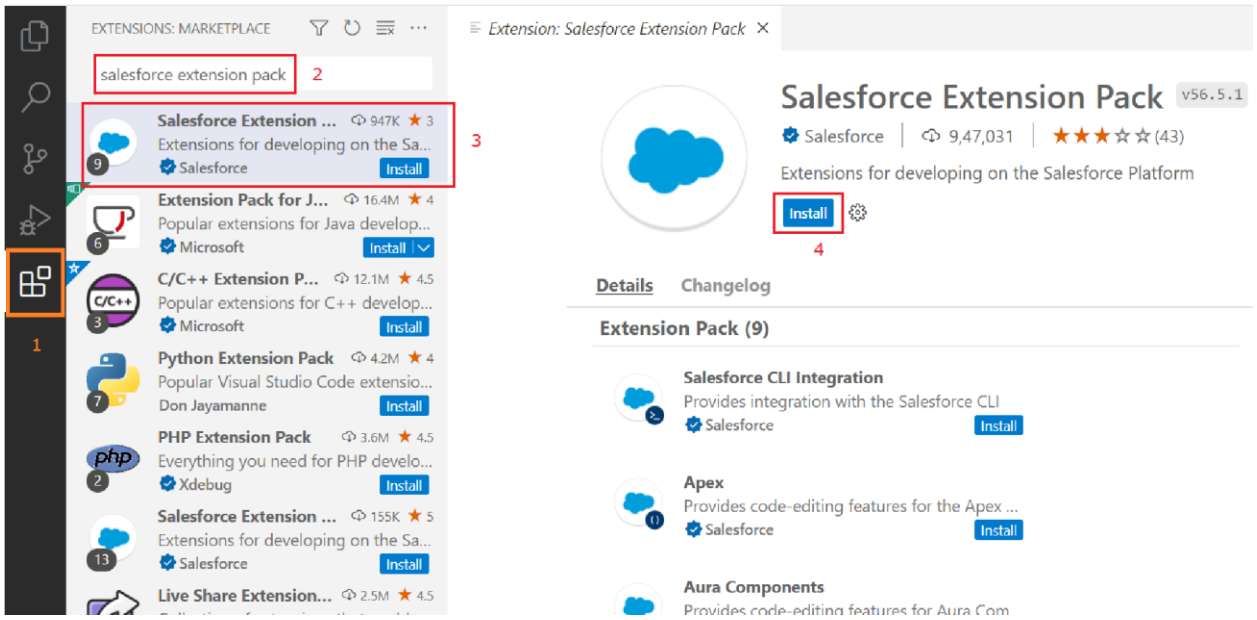
Download the version of the software that is compatible with your operating system and install it.

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

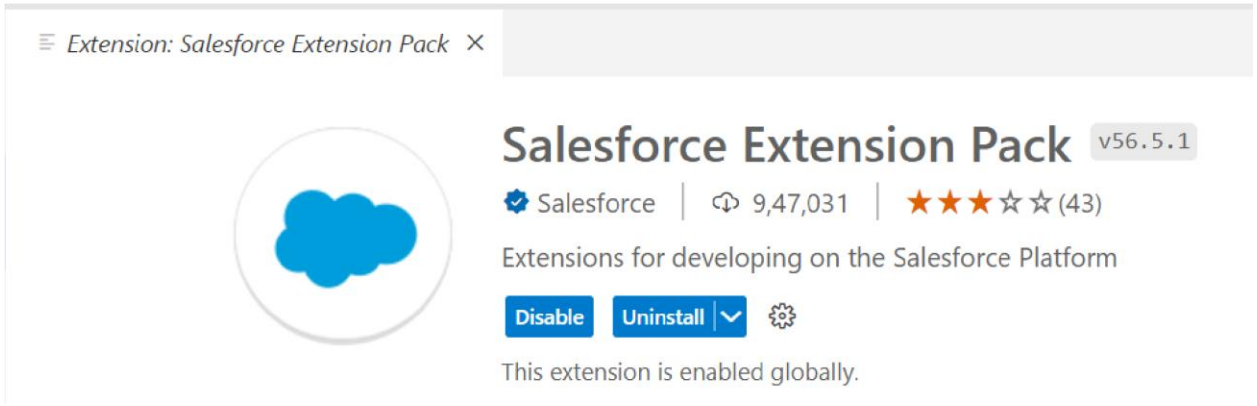
Activity- 3: Install the Salesforce Extension Pack

In the VS Code,

1. go to extensions (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.



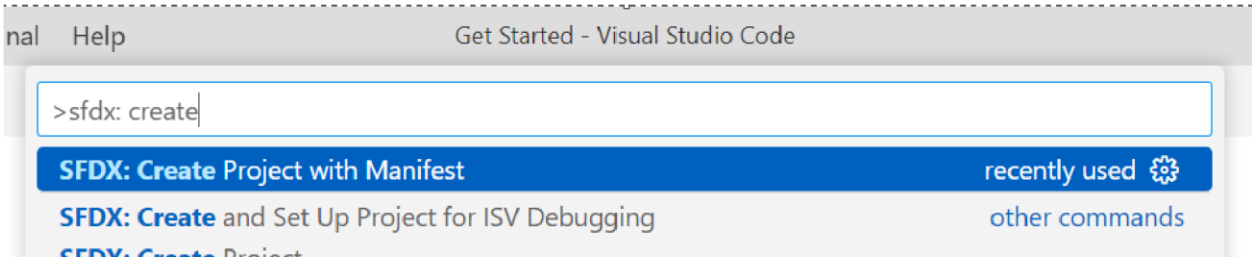
The extension pack is installed successfully



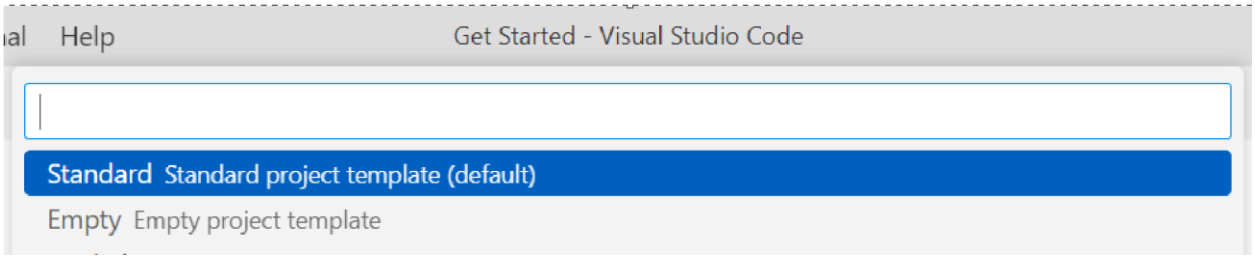
Install the Salesforce Extension Pack

Activity- 4 : Create a project in VS Code

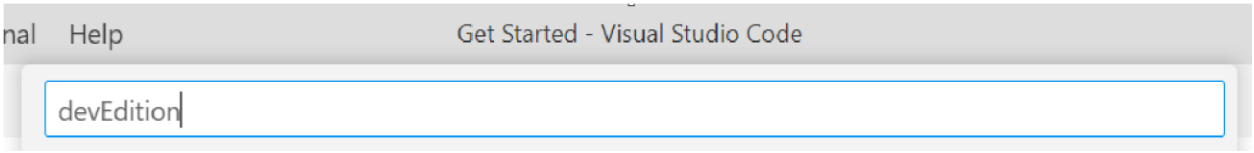
1. Press CTRL + SHIFT + P, type sfdx: create
2. select SFDX: Create Project with Manifest



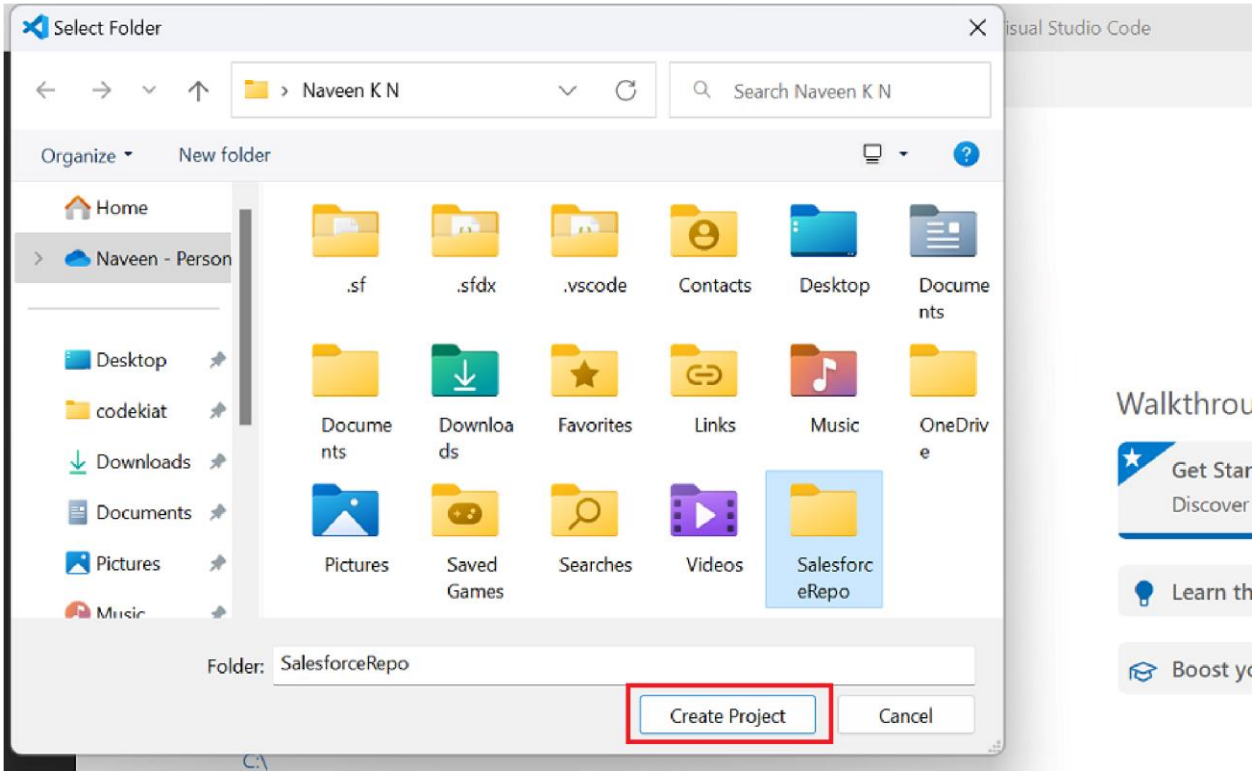
3. Select the Standard project template



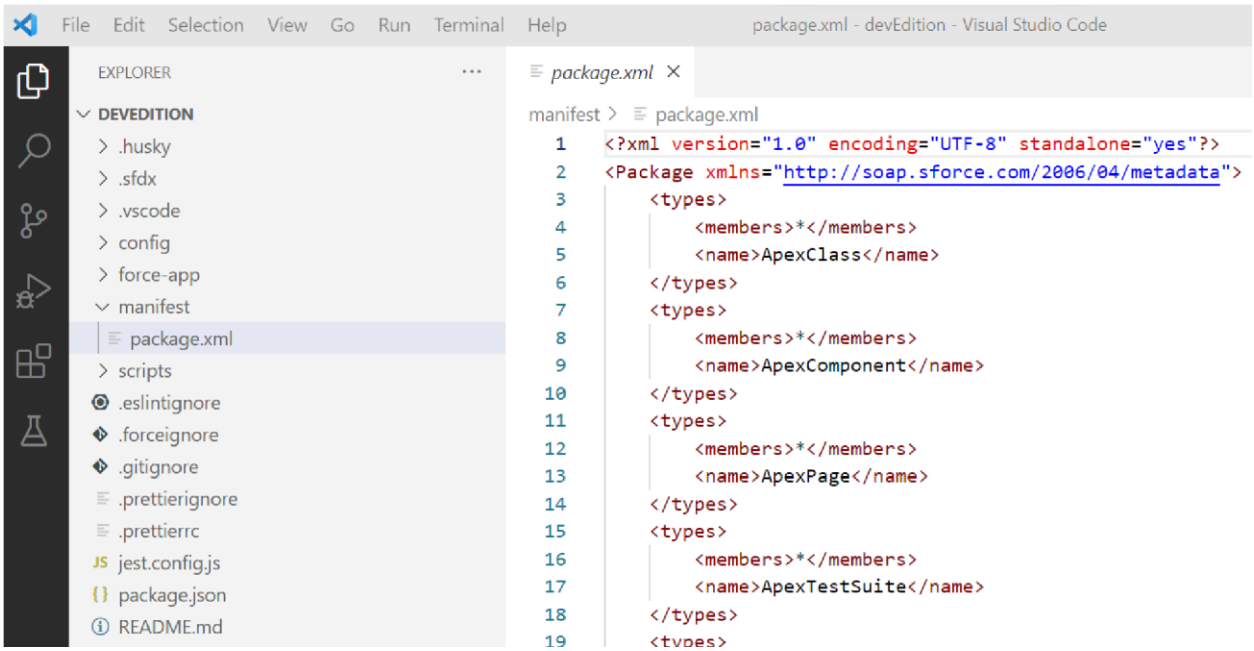
4. Type a project name and Click Enter.



5. Select the folder (create a new folder if required) and click Create Project



6. The new project is created with package.xml



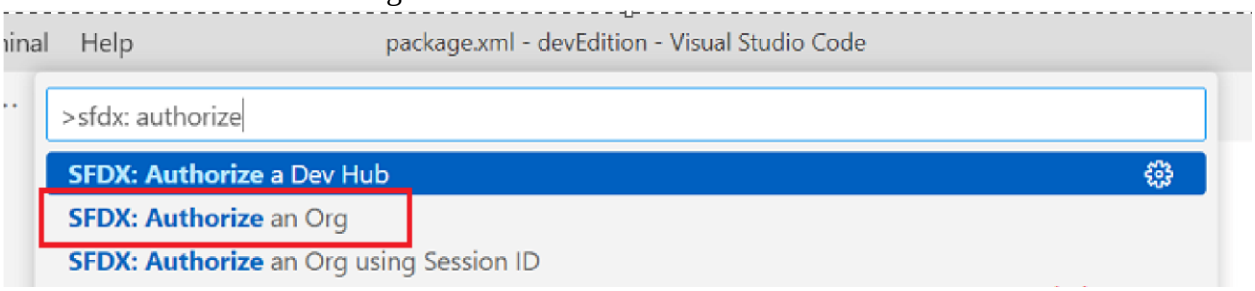
Default Package.xml contains various metadata types. I have updated Package.xml as shown below as we deal only with LWC in this article.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>LightningComponentBundle</name>
    </types>
    <version>55.0</version>
</Package>
```

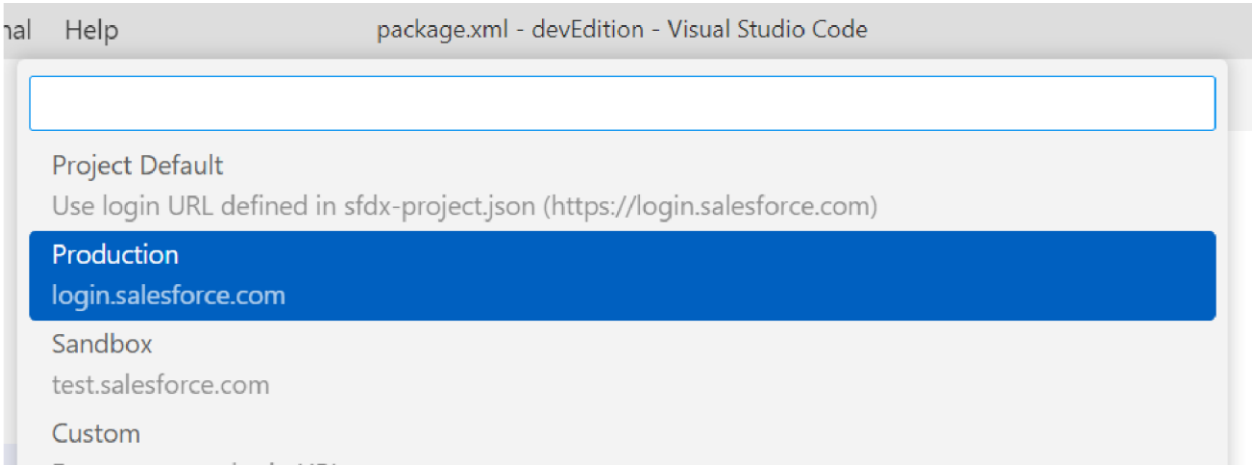
Activity- 5 : Authorize an org

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

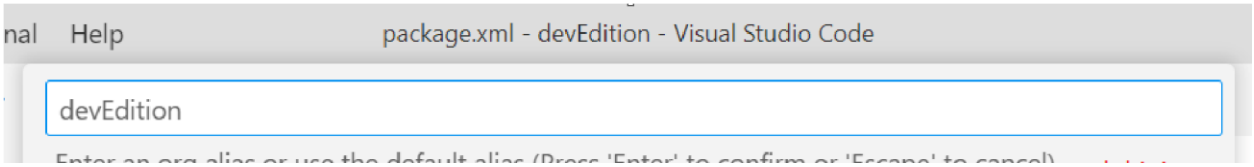
1. Press CTRL + SHIFT + P, type sfdx: authorize.
2. select SFDX: Authorize an Org from the list



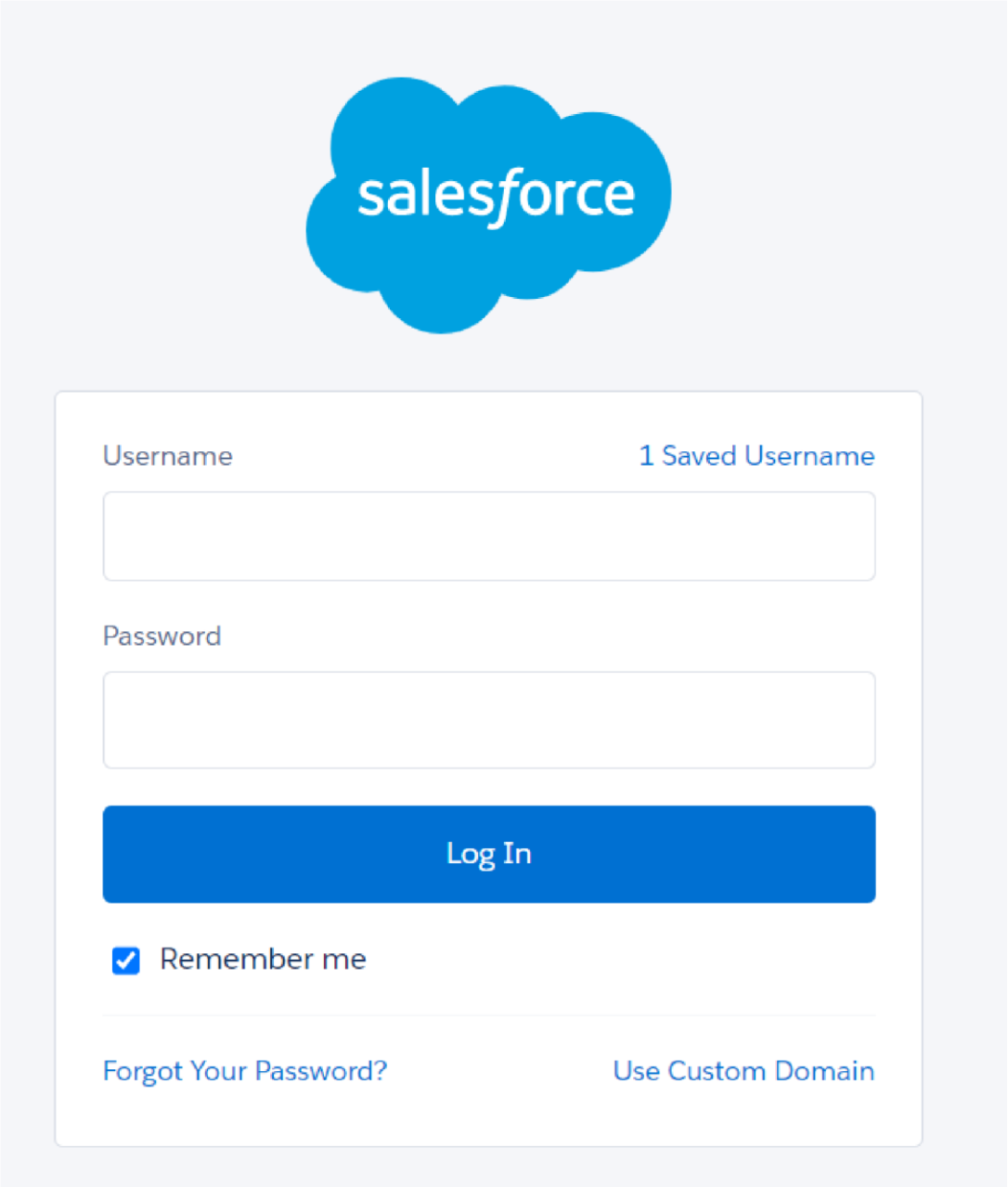
3. Choose your Salesforce instance.
   For developer edition and production instances select Production.



4. For this demonstration, I used the developer edition, hence it is Production.
5. Give a project name and press Enter



6. The Salesforce login page opens in the browser.
7. Enter the credentials and click Log In

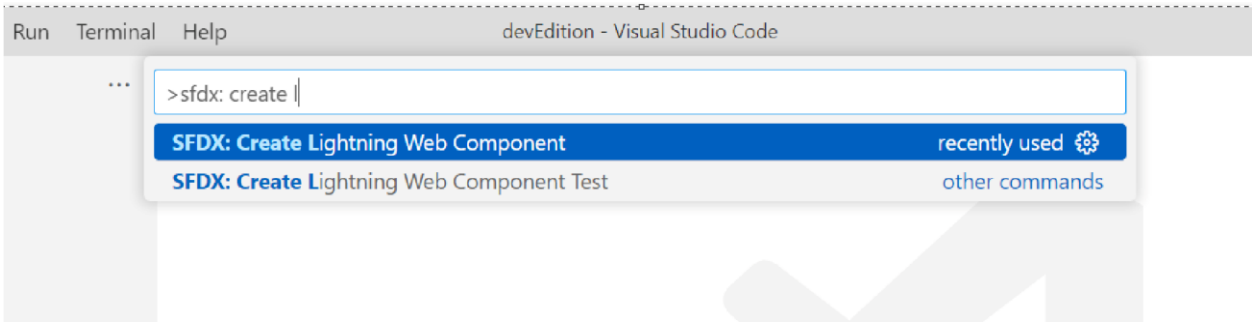


8. It will be successfully authorized.

Activity- 6 : Create Lightning Web Component

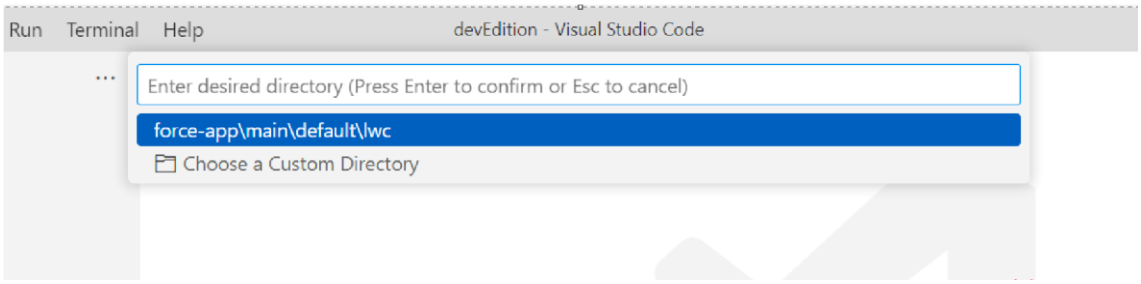
XML File :

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

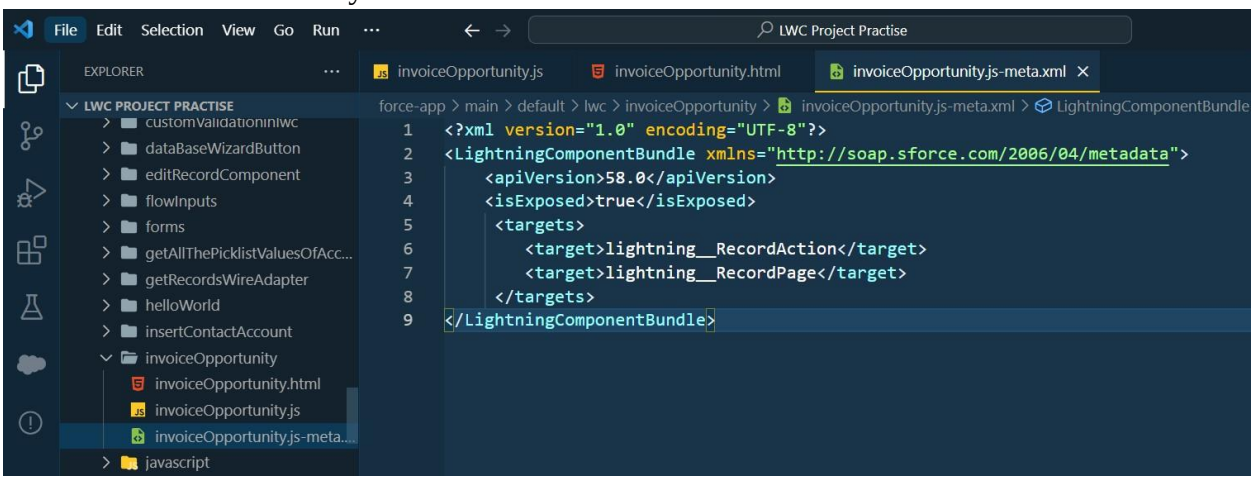


2. Give the name “InvoiceOpportunity” and press Enter.

3. Choose the directory.



4. LWC is created successfully.



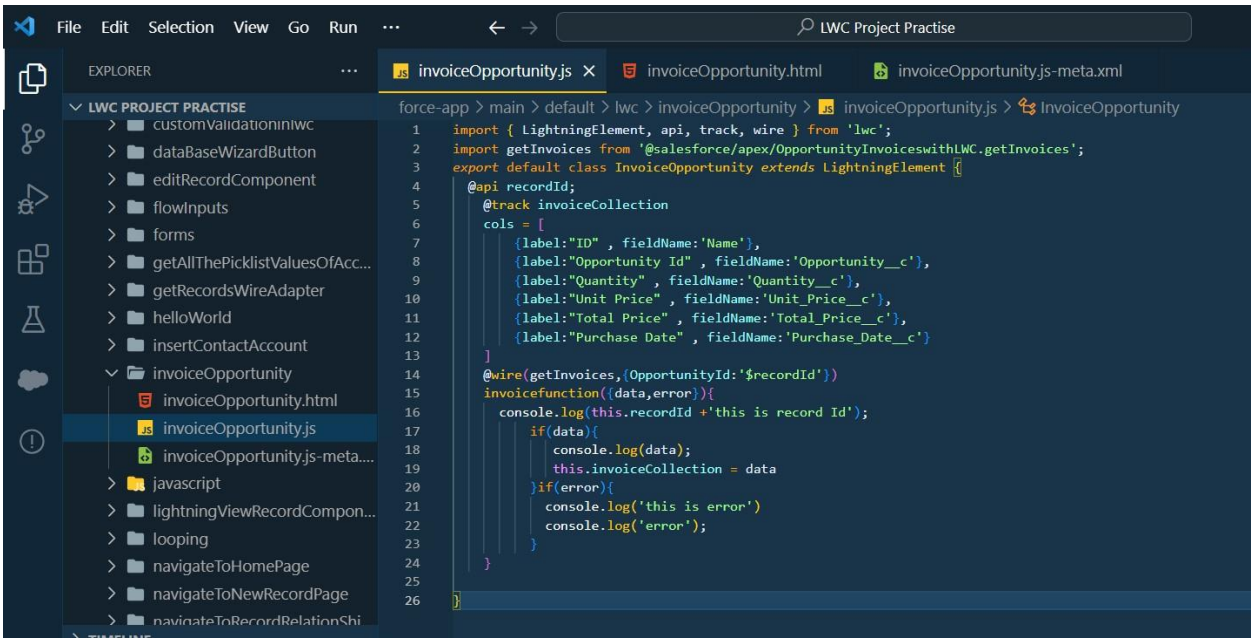
5. Copy and paste the below-mentioned code in the InvoiceOpportunity.js-meta.xml and update the apiVersion tag with the latest API version.

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 :

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



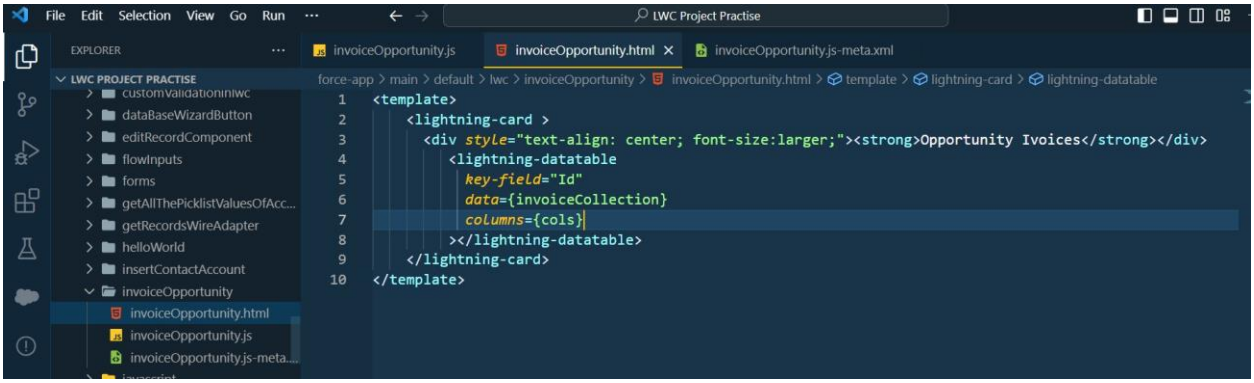
JS File Code :

```
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');
          }
    }
}
```

HTML File :

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



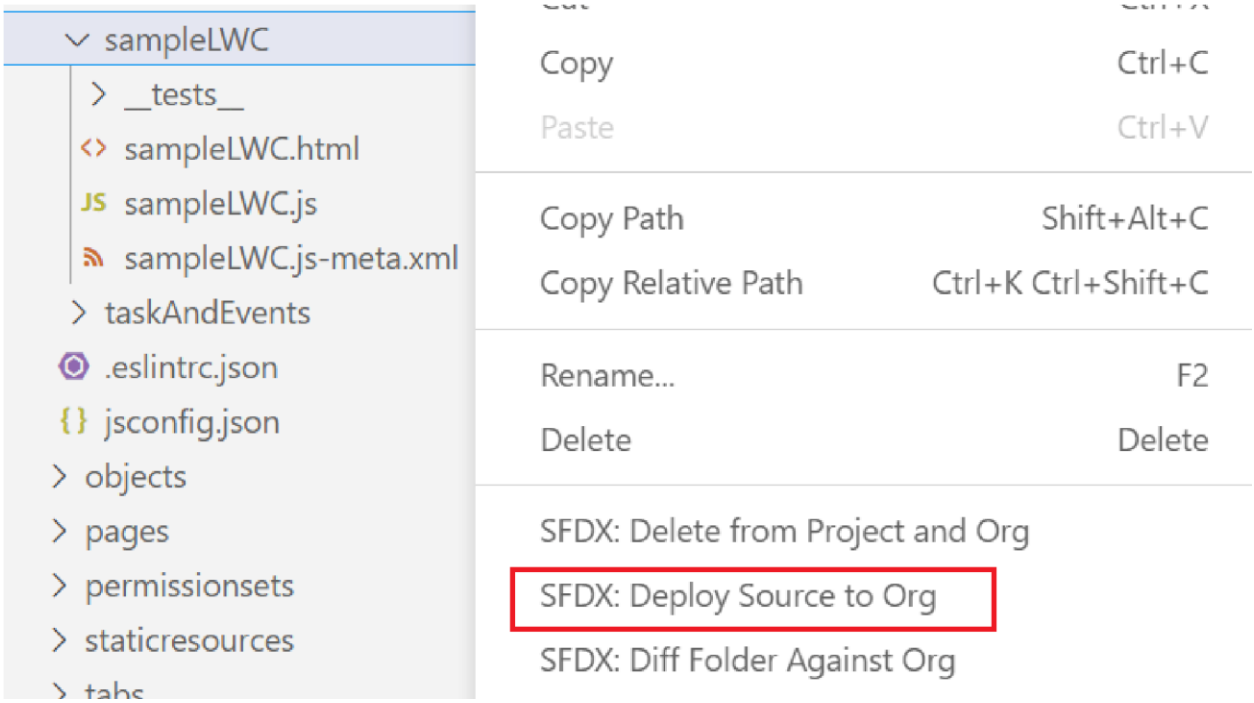
HTML File Code:

```
<template>
```

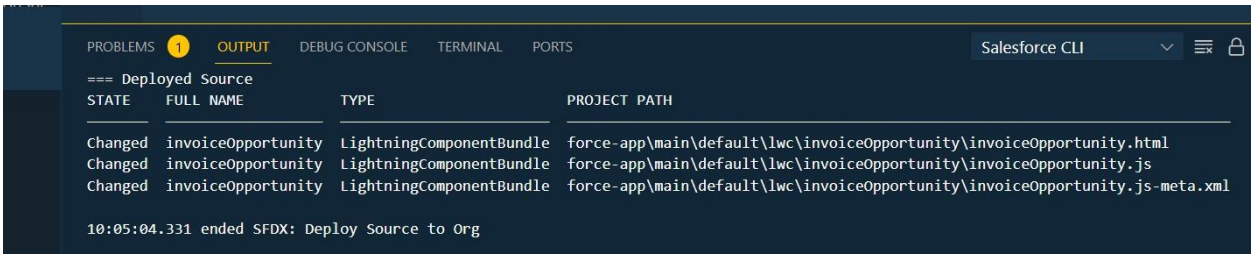
```
    <lightning-card >
      <div style="text-align: center; font-size:larger;"><strong>Opportunity
Ivoices</strong></div>
        <lightning-datatable key-
          field="Id"
          data={invoiceCollection}
          columns={cols}
        ></lightning-datatable>
    </lightning-card>
</template>
```

Deploy Component:

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

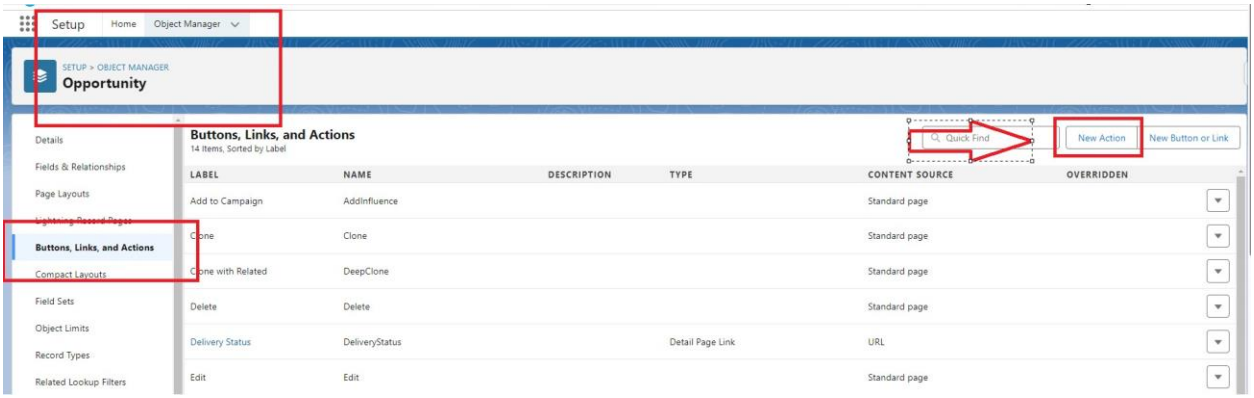


2. Once the deployment is complete, you will see the below-highlighted message in the output tab

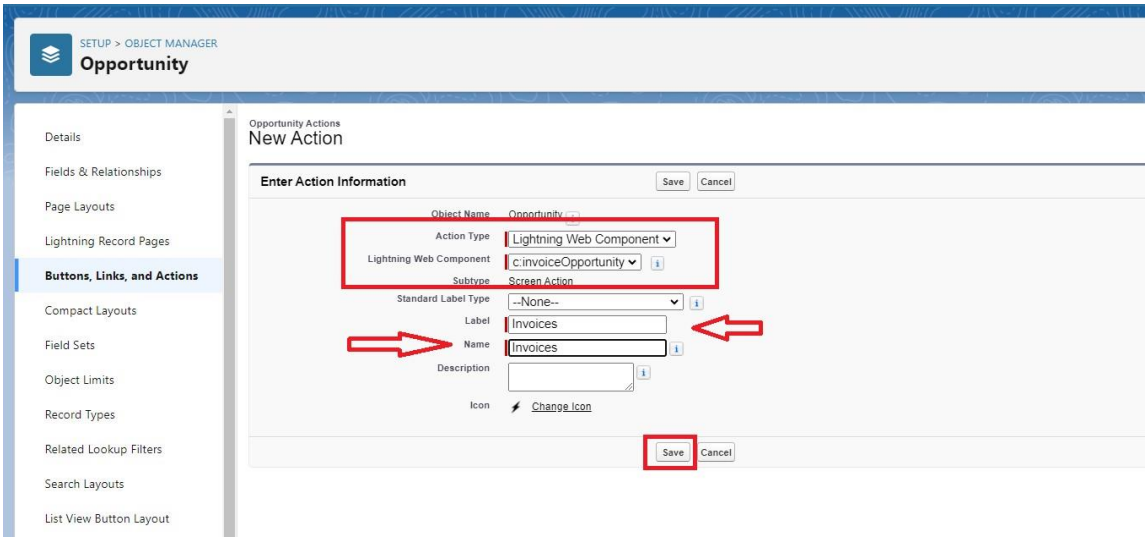


Activity- 7 : Create Button to Add on Opportunity

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.



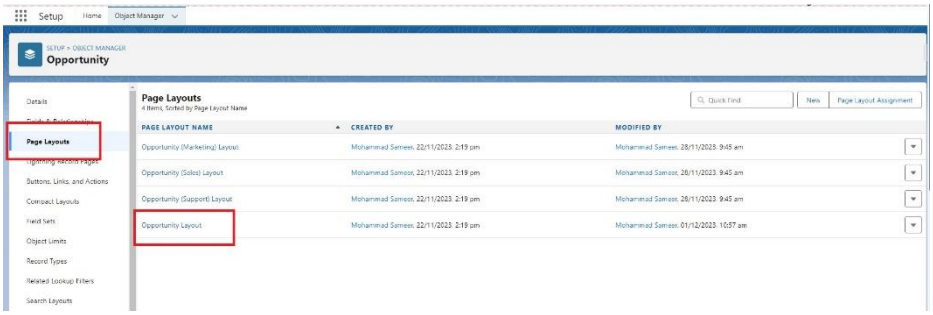
6. Select Action type as Lightning Web Component
7. Select the InvoiceOpportunity component
   a. Label :- Invoices
   b. Name :- Invoices
8. As given on below image



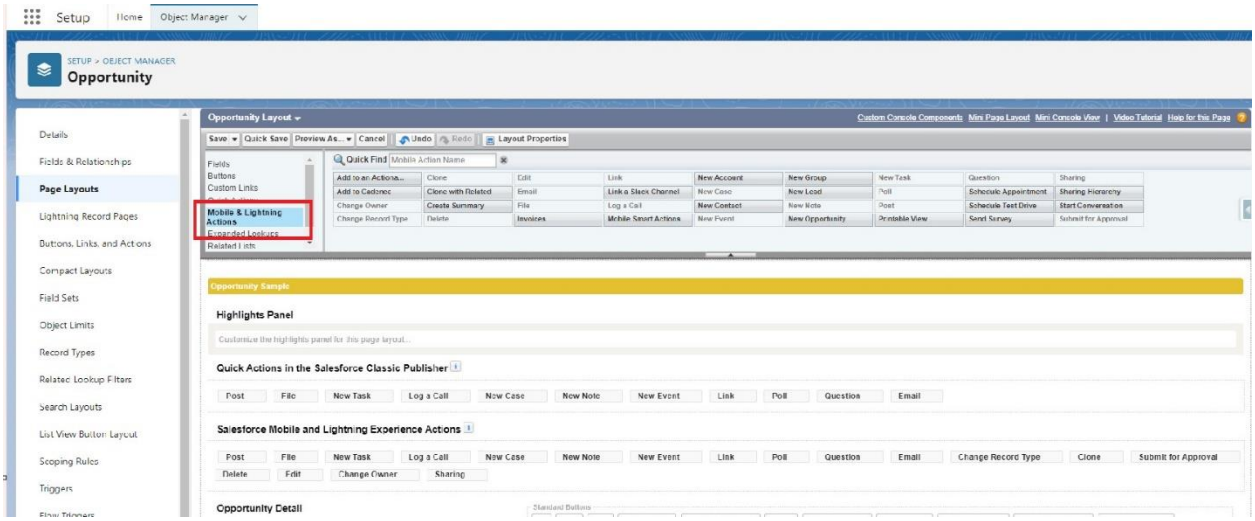
9. Click on Save and your action Button is Ready.

Activity- 8 : Add InvoiceOpportunity into Opportunity Record Page

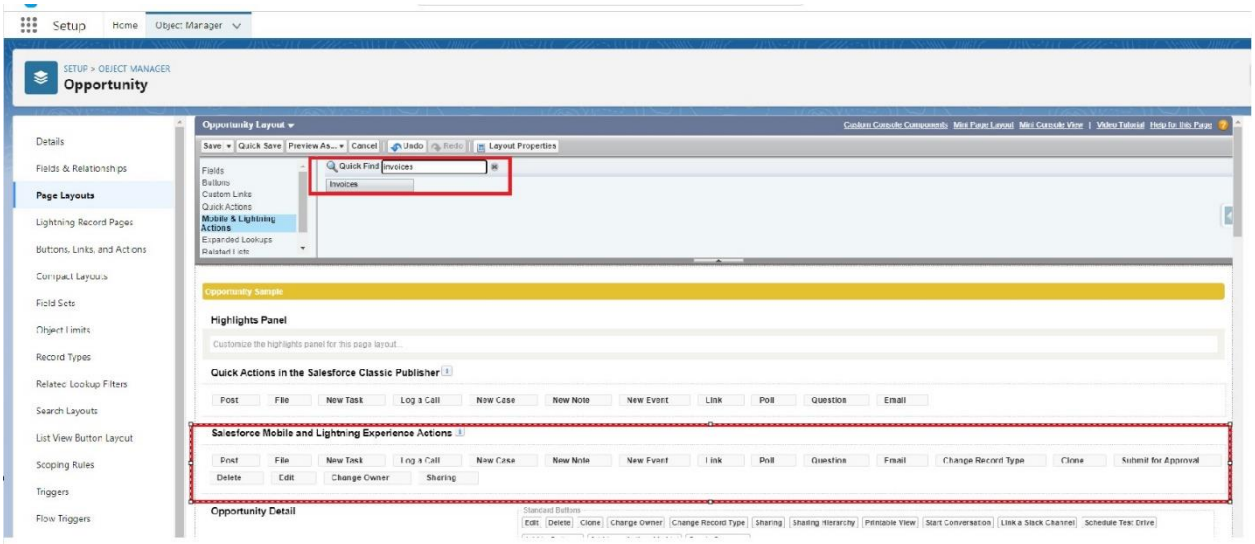
1. On Opportunity Object Manager Click on Page layout.
2. Click on OpportunityLayout.



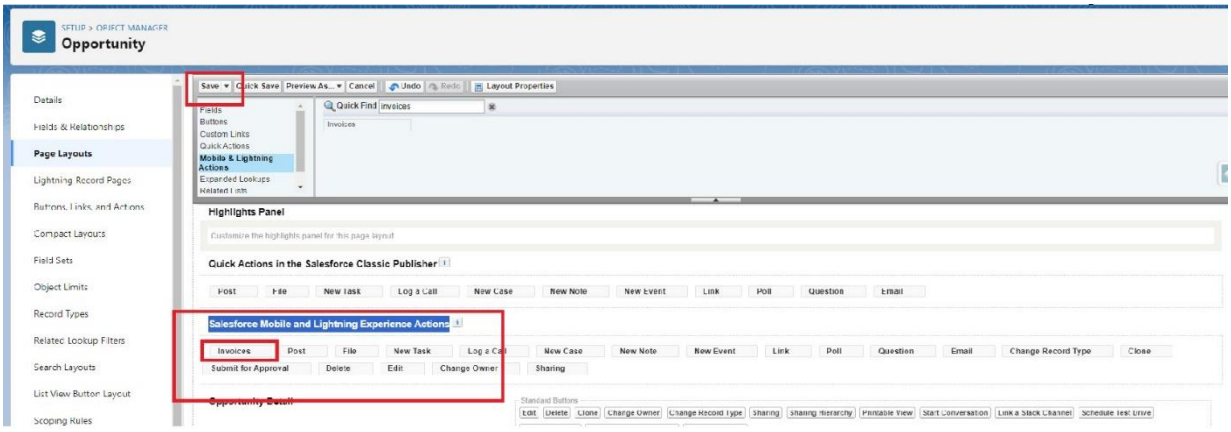
3. Click on Mobile And Lightning Action as show on below Image



4. Search for invoice on Quick Find.



5. Drag and Drop the Invoice into Salesforce Mobile and Lightning Experience Action



6.Click on Save.

Milestone 9 : Apex Schedulers :

The Apex Scheduler lets you delay execution so that you can run Apex classes at a specified time. This is ideal for daily or weekly maintenance tasks using Batch Apex. To take advantage of the scheduler, write an Apex class that implements the Schedulable interface, and then schedule it for execution on a specific schedule.

Schedulable Apex Syntax :

To invoke Apex classes to run at specific times, first implement the Schedulable interface for the class. Then, schedule an instance of the class to run at a specific time using the System.schedule() method.

After you implement a class with the Schedulable interface, use the System.schedule() method to execute it. The System.schedule() method uses the user's timezone for the basis of all schedules, but runs in system mode—all classes are executed, whether or not the user has permission to execute the class.

SYNTAX :

```
public class SomeClass implements Schedulable {
   public void execute(SchedulableContext ctx) {
      // awesome code here
   }
}
```

Objective :

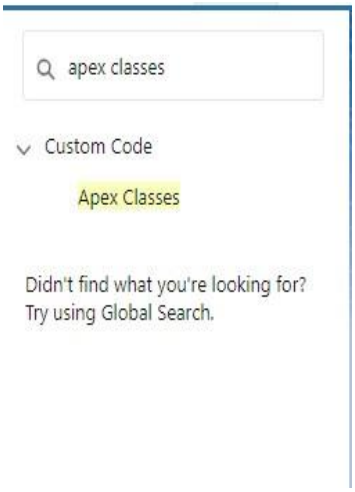
- Through this schedulable class, we can see all the Closed Lost Opportunities.
- We can delete all the Closed lost Opportunities by this Scheduled method on every monday as weekly.
  1) Login to the respective account and navigate to the gear icon in the top right corner.
  2) Click on the Developer 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 ”

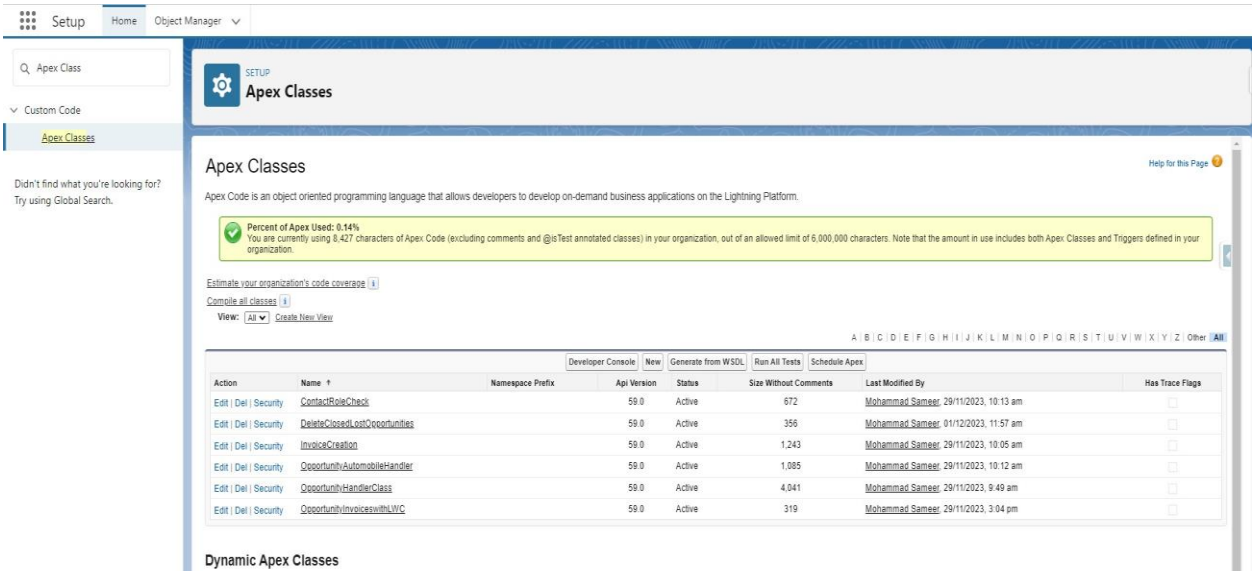
CODE SNIPPET :


```
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;
      }
   }
}
```

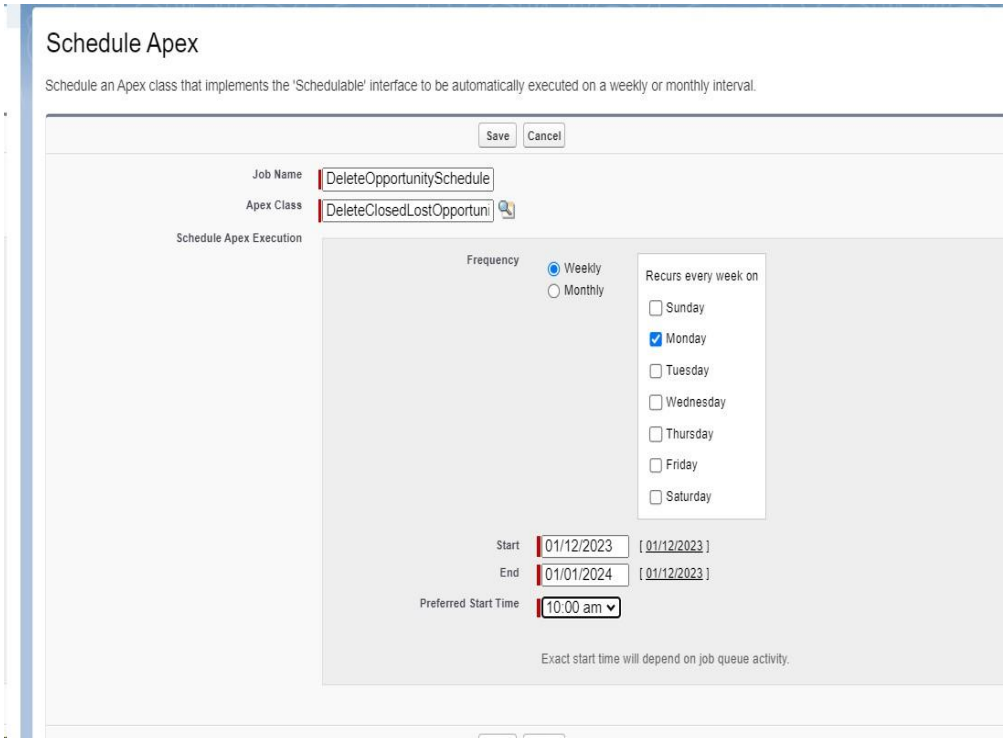
Schedule the Apex class:

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

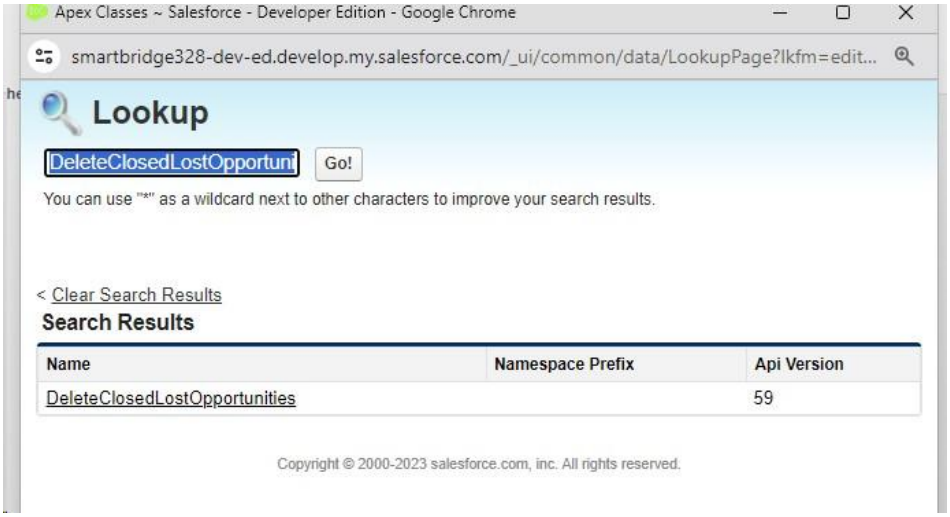




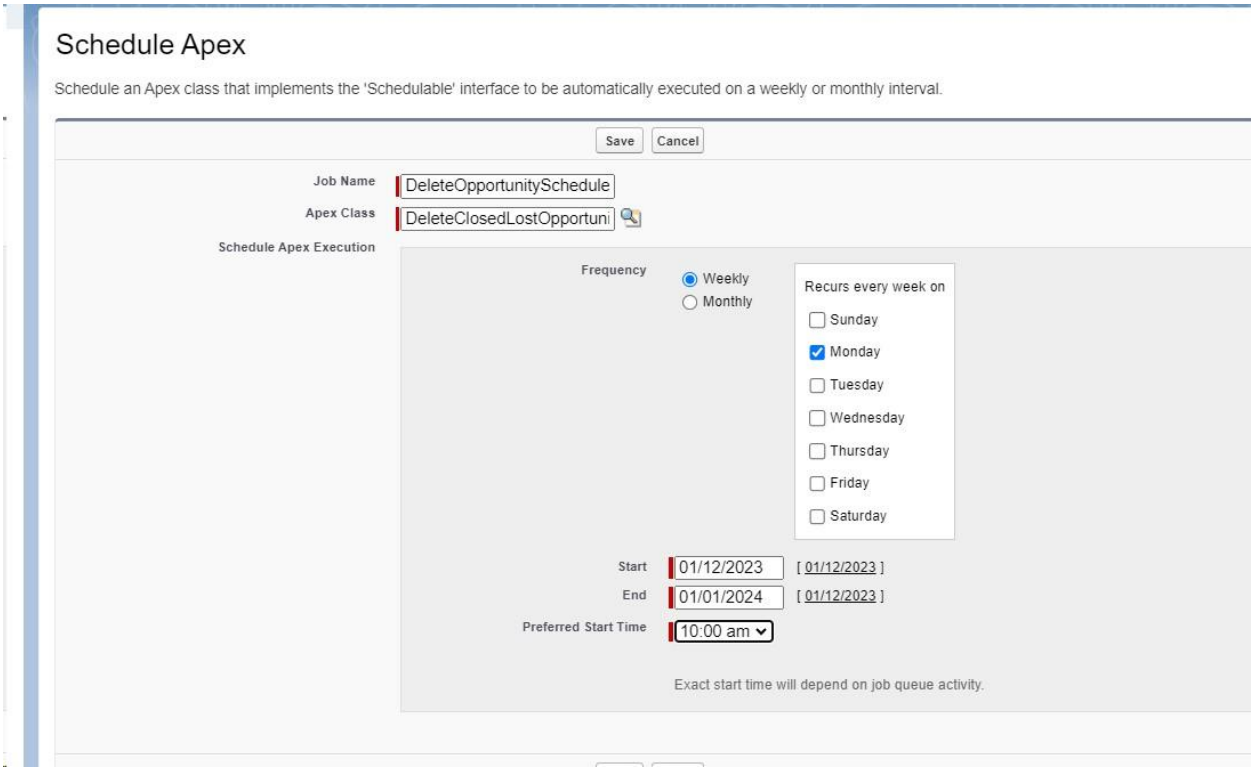
- Click on Schedule Apex and enter the Job name.
  - Job Name : DeleteOpportunitySchedule



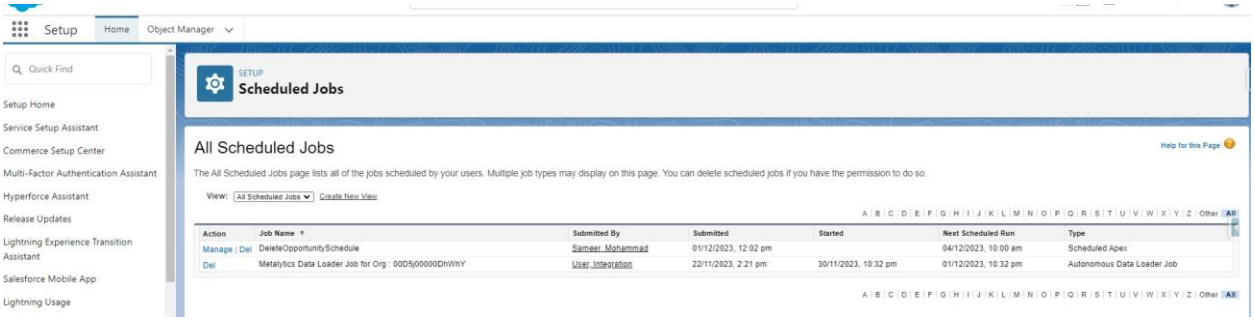
1) Now click on the search icon present near the Apex class : Goto the Lookup icon beside → click on it → select DeleteClosedLostOpportunities.



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



3) 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.

Milestone 10 : 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.

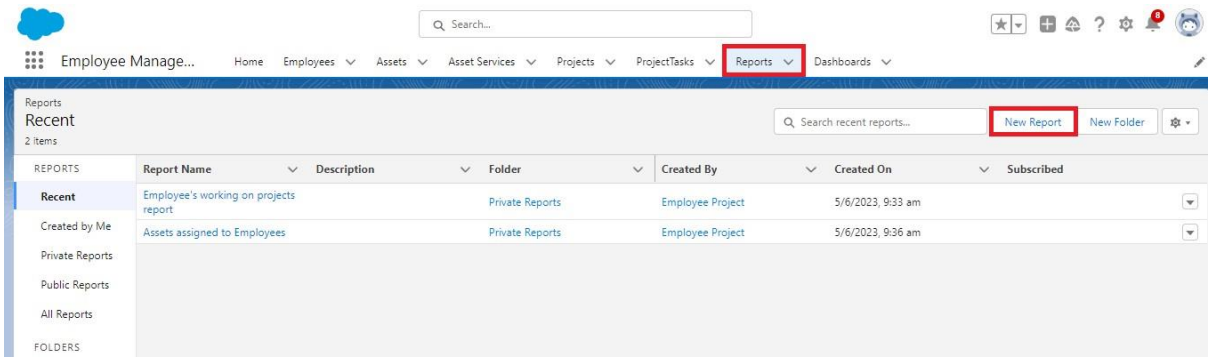
Types of Reports in Salesforce

1. Tabular
2. Summary
3. Matrix
4. Joined Reports Use Case:

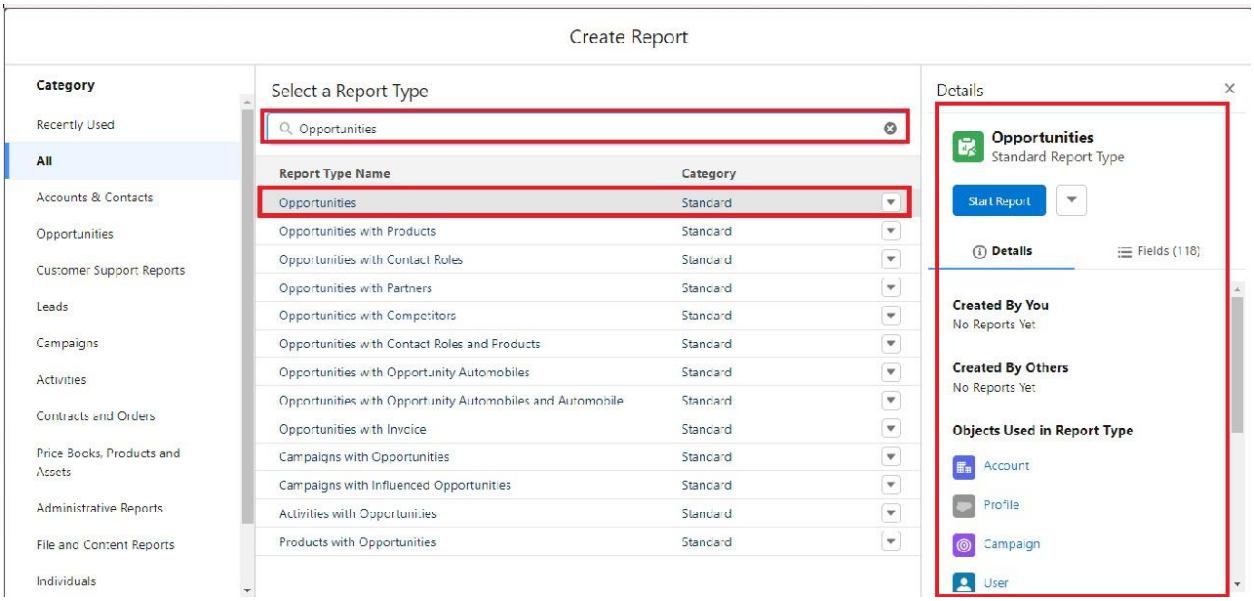
The CEO of an organization wants to have brief data on Opportunity Sales along with Invoices generated.
Let's create a Report.

Activity- 1 : Create Report on Opportunity

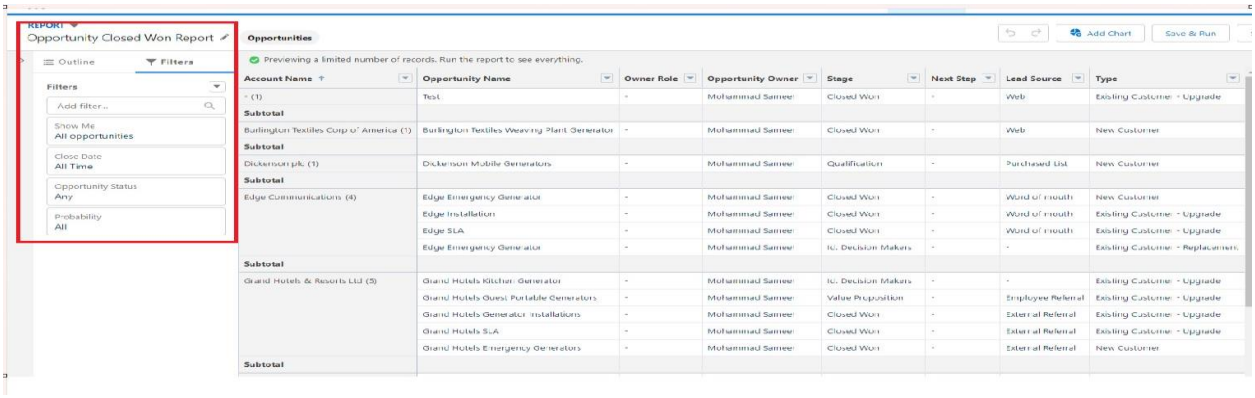
1. Go to the app → click on the reports tab
2. Click New Report.

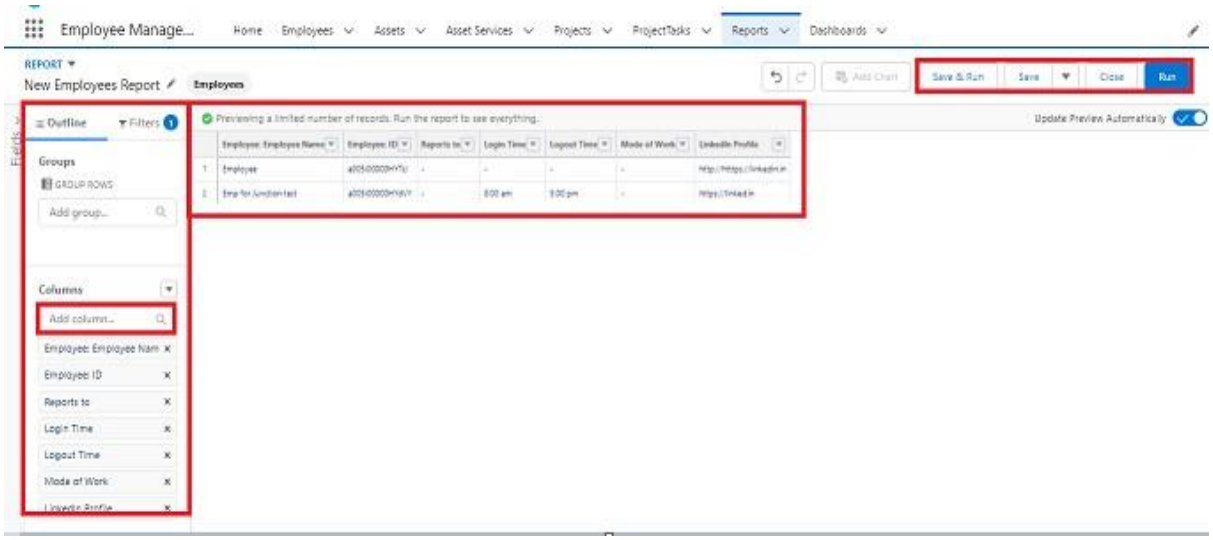


3. Select report type from category or from report type panel or from search panel → click on start report.



4. Customize your report
   - Add fields from left pane as shown below





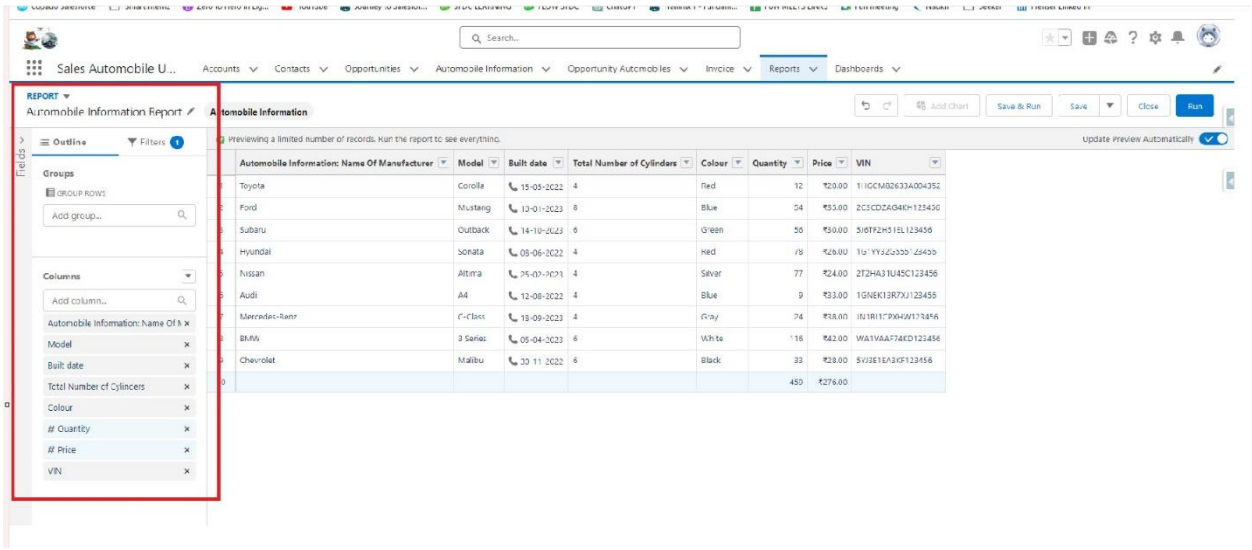
Add the Above Filter as well.

5. Save or run it.

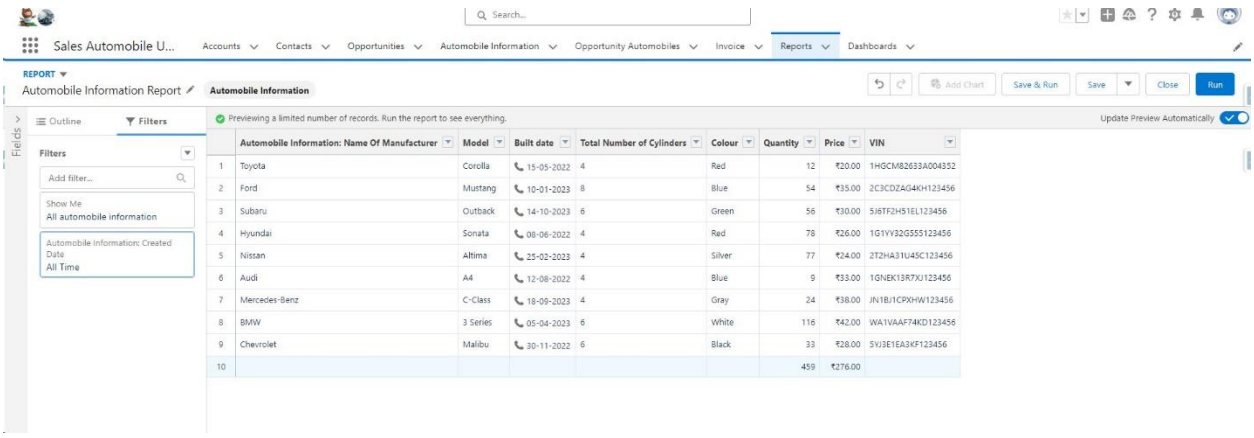
Note: Reports may get varied from the above pictures as the data might be different.

Activity- 2 : Create Report on Automobile Information.

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



Filters :-



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

Milestone 11 :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.

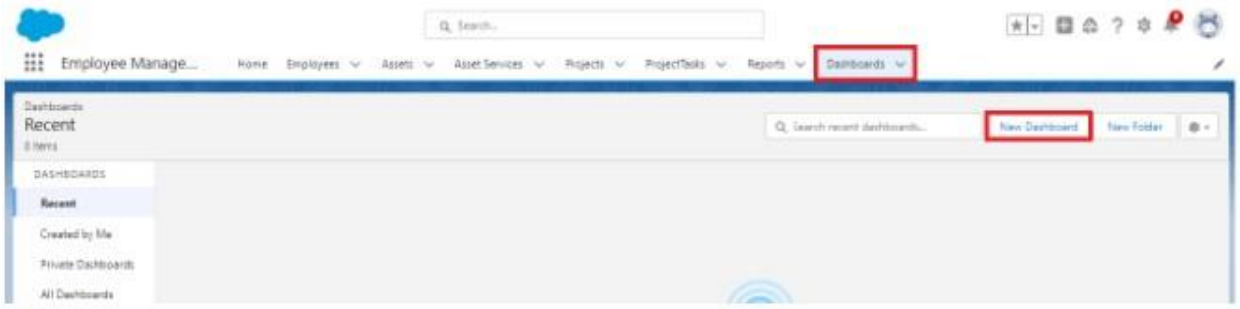
Use Case:

As an Admin for the organization you keep pushing yourself to reach out the business requirements to take the organization to peak heights and all your superiors are very much impressed with your efforts and work dedication. In addition with reports you make an ease for the CEO in viewing the reports with data visualization. So he doesn't have to search for the data he wants during the meetings.

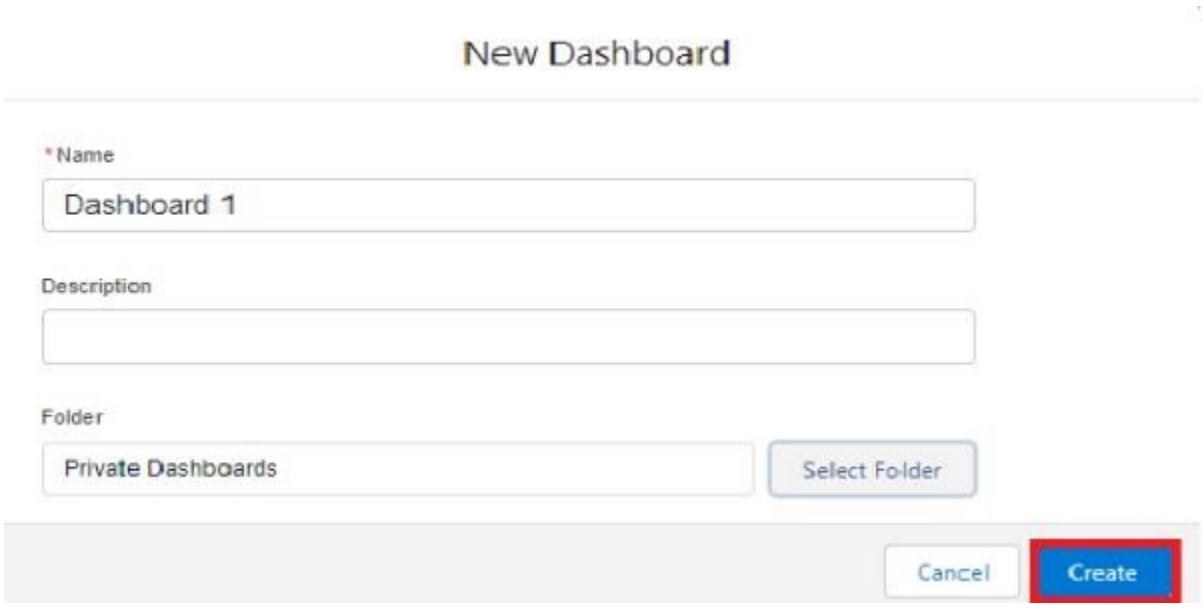
Activity 1:

Create Dashboard

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

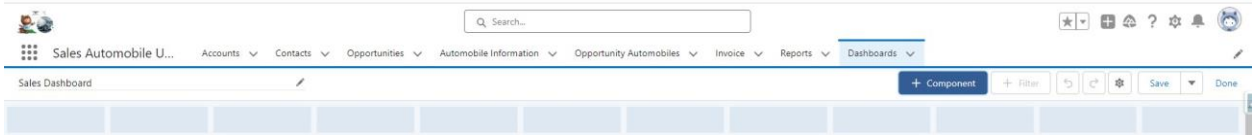


2. Give a Name and click on Create.

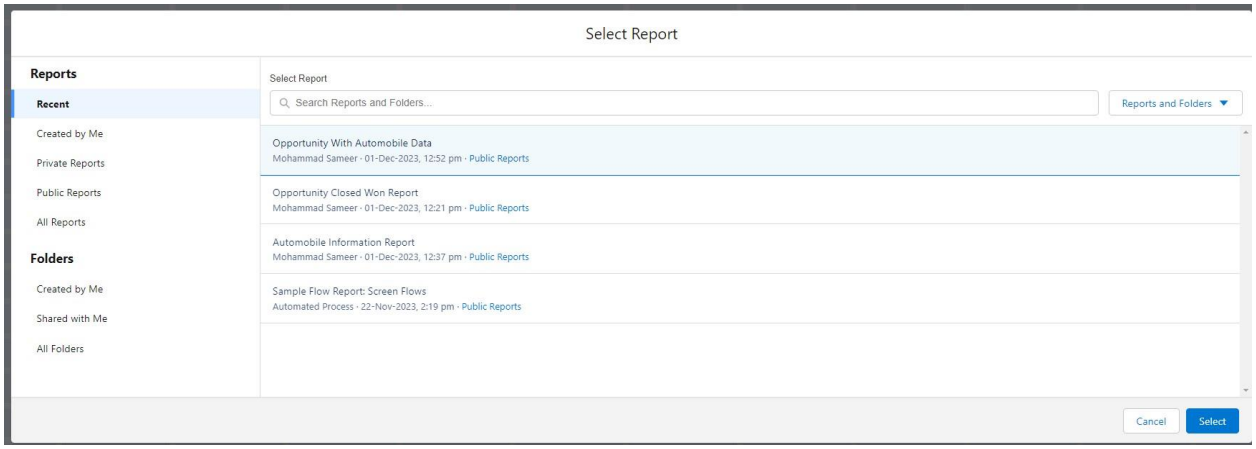


Name : Automobile Sales

3. Select add component.

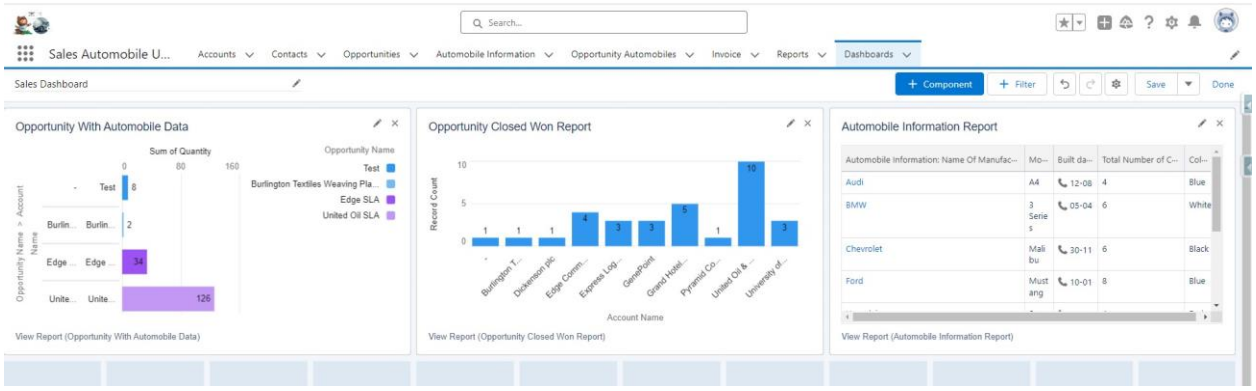


4. Select a Report and click on select.



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

The Created Dashboard will look like this.



5.Testing and Validation:

- **Unit Testing**:
  - ❖ Conducted tests for Apex classes and triggers to verify accuracy and reliability.
- **User Interface Testing**:
  - ❖ Ensured that all UI components function correctly across devices and roles.

6.Key Scenarios Addressed by Salesforce in the Implementation Project:

- ❖ Lead-to-Opportunity Conversion: Streamlining the process of qualifying leads and moving them through the sales pipeline
- ❖ Automated Follow-Ups: Scheduling automated reminders for follow-ups to enhance customer engagement.
- ❖ Customer Insights: Providing actionable insights into customer preferences and buying patterns through analytics.
- ❖ Inventory Management: Monitoring and updating automobile inventory in real-time.

7. Conclusion:

The project successfully implemented a robust CRM system tailored to the needs of the automobile sales domain. It enhanced sales productivity, streamlined customer interactions, and provided insightful analytics for informed decision-making. By leveraging Salesforce, the project achieved its goal of transforming the automobile sales process into a more efficient and customer-centric system.

**********END**********