

## **Sales Automobile Using Salesforce CRM**

### **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.

### **What you'll learn**

1. Real Time Salesforce Project
2. Data Modelling
3. Creating an Application
4. User Interface Customization
5. Object & Relationship in Salesforce
6. Formula fields and Validation rules.
7. Conditional formatting.
8. Reports & Dashboards.
9. Apex.
10. LWC.

### **System Requirements:**

1. Windows 8 machine
2. Installed with two web browser
3. Bandwidth of 30mbps

### **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.

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

The image consists of two side-by-side screenshots. The left screenshot is a promotional graphic for Salesforce development. It features a blue background with a white computer monitor in the center displaying a complex flowchart or application interface. Above the monitor, there are several small, glowing blue and white circular icons arranged in a curved path. Below the monitor, the text reads: "Build enterprise-quality apps fast to bring your ideas to life". Underneath this, there is a bulleted list of features: "Build apps fast with drag and drop tools", "Customize your data model with clicks", "Go further with Apex code", "Integrate with anything using powerful APIs", "Stay protected with enterprise-grade security", and "Customize UI with clicks or any leading-edge web framework". The right screenshot is a screenshot of a web browser showing the "Sign up for your Salesforce Developer Edition" page. At the top, it says "Sign up for your Salesforce Developer Edition" and "A full-featured copy of the Platform, for free". Below this is a text box containing the instruction: "Complete the form to start your free trial. Our team will be in touch to help you make the most of your trial." The form itself has five fields: "First Name\*" with input field "Your first name", "Last Name\*" with input field "Your last name", "Email\*" with input field "Your email address", "Role\*" with dropdown menu "Your job role", and "Company\*" with input field "Company Name".

- 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

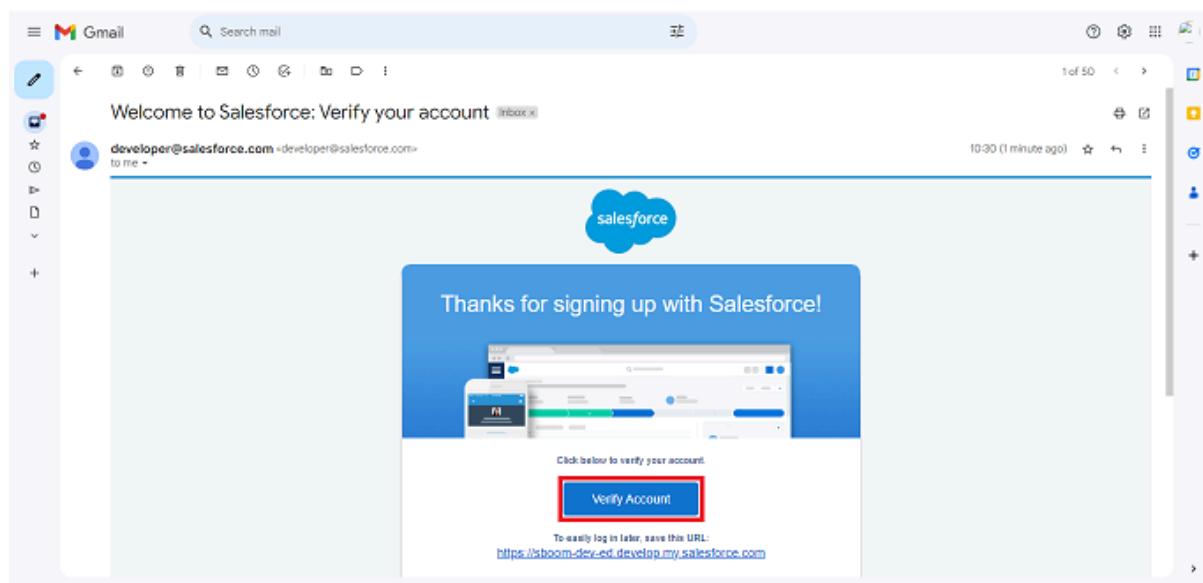
This need not be an actual email id, you can give anything in the format :

[username@organization.com](mailto:username@organization.com)

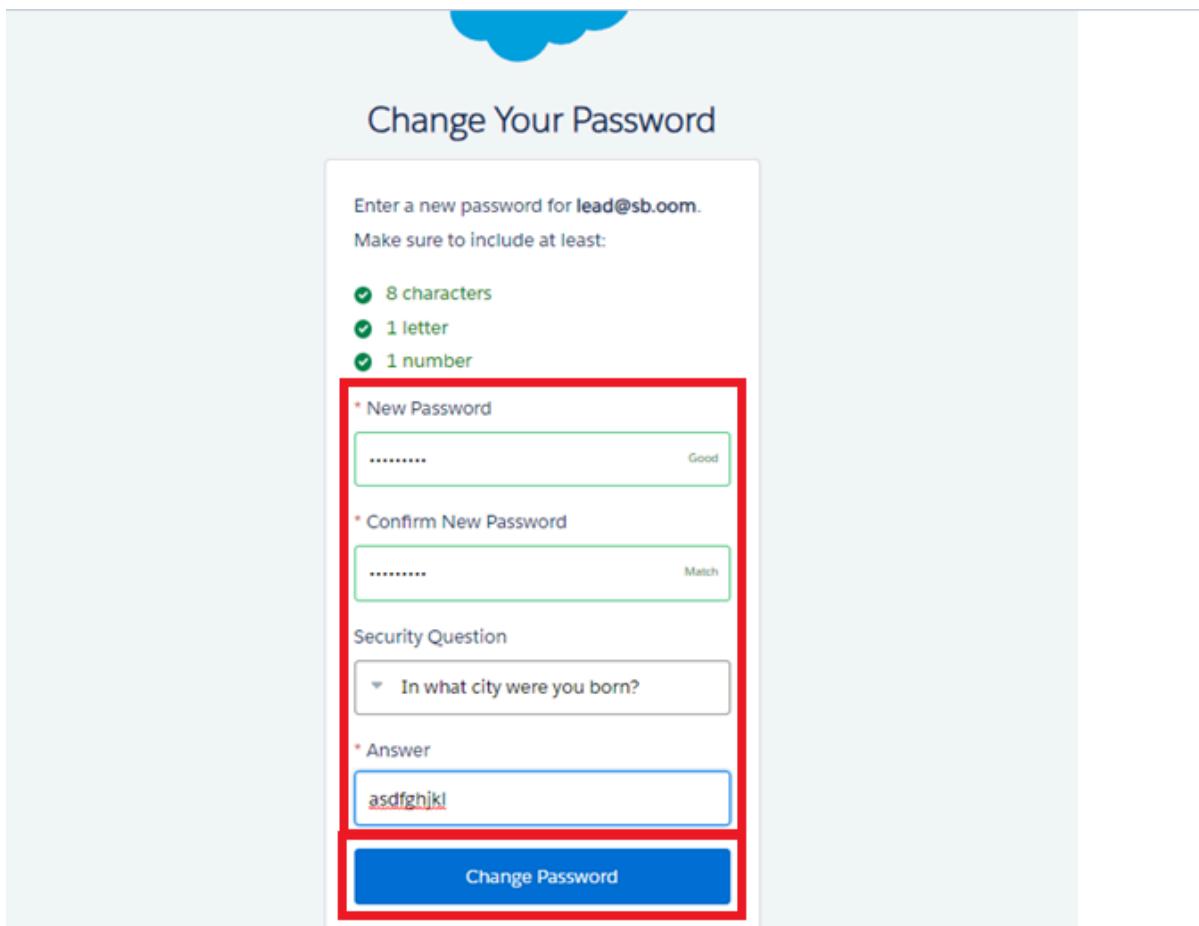
Click on sign me up after filling these.

### **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.

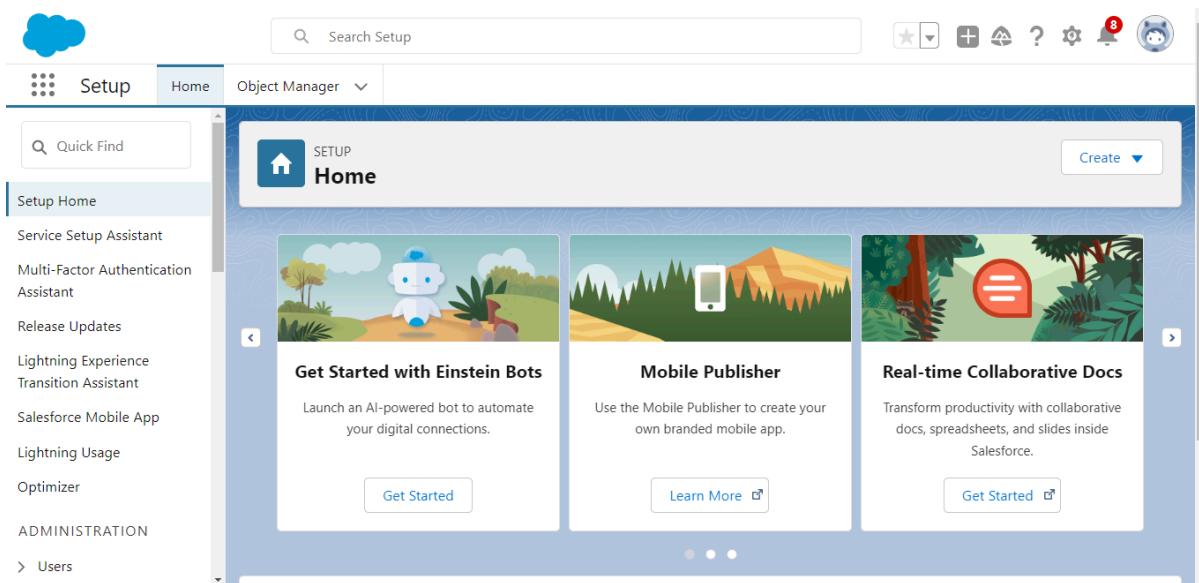


2. Click on Verify Account
3. Give a password and answer a security question and click on change password.



The screenshot shows the "Change Your Password" page in the Salesforce setup interface. At the top, it says "Change Your Password". Below that, it asks to enter a new password for "lead@sb.oom" and lists requirements: "8 characters", "1 letter", and "1 number". A red box highlights the "New Password" and "Confirm New Password" fields, which both show "Good" and "Match" respectively. Below these are "Security Question" and "Answer" fields, with "In what city were you born?" and "asdfghjkl" respectively. A red box also highlights the "Change Password" button at the bottom.

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



The screenshot shows the Salesforce Setup Home page. The top navigation bar includes "Setup", "Home", and "Object Manager". The main content area features a "Home" section with three cards: "Get Started with Einstein Bots", "Mobile Publisher", and "Real-time Collaborative Docs". The sidebar on the left contains links such as "Setup Home", "Service Setup Assistant", "Multi-Factor Authentication Assistant", "Release Updates", "Lightning Experience Transition Assistant", "Salesforce Mobile App", "Lightning Usage", "Optimizer", and "ADMINISTRATION".

## **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:**

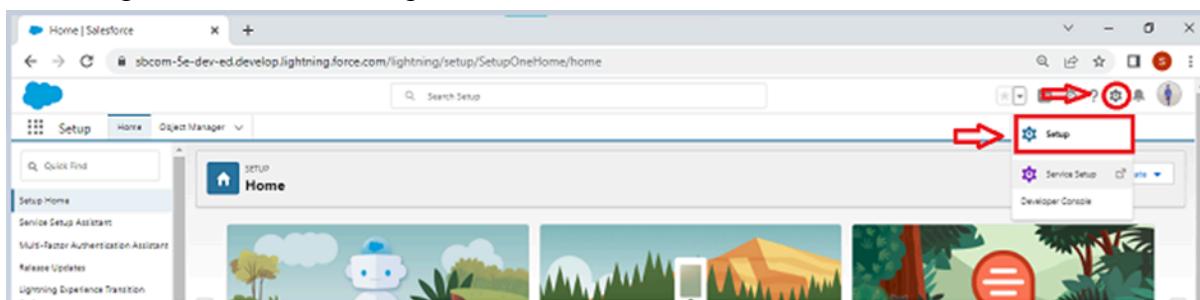
1. **Standard Objects:** Standard objects are the kind of objects that are provided by salesforce.com such as users, contracts, reports, dashboards, etc.
2. **Custom Objects:** Custom objects are those objects that are created by users. They supply information that is unique and essential to their organization.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

Define object and fields

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

**CSV File Details**

Encoding Format: Unicode (UTF8)    Values Separated By: Comma    Field Label Source: Enter manually    Field Labels Row: 1    Import 0 rows of Data? No, skip import Yes, import data

**Fields 9 of 9 to import**  Hide mapped fields

IMPORT FILE FIELD NAME	SALESFORCE FIELD NAME	SALESFORCE FIELD TYPE	ADD TO LAYOUTS	FIELD PREVIEW
Name Of Manufacturer	Name Of Manufacturer	Text	<input checked="" type="checkbox"/>	Ioyota
Model	Model	Text	<input checked="" type="checkbox"/>	Corolla
Engine Number	Engine Number	Text	<input checked="" type="checkbox"/>	ABC12345
VIN	VIN	Text	<input checked="" type="checkbox"/>	1HGCM8263A004352
Total Number of Cylinders	Total Number of Cylinders	Picklist	<input checked="" type="checkbox"/>	4
Colour	Colour	Picklist	<input checked="" type="checkbox"/>	Red
Built date	Built date	Phone	<input checked="" type="checkbox"/>	15-05-2022
Price	Price	Currency	<input checked="" type="checkbox"/>	520
Quantity	Quantity	Integer	<input checked="" type="checkbox"/>	12

**Back** **Next**

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.

Define object and fields

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

**CSV File Details**

Encoding Format: Unicode (UTF8)    Values Separated By: Comma    Field Label Source: Enter manually    Field Labels Row: 1    Import 0 rows of Data? No, skip import Yes, import data

**Fields 5 of 5 to import**  Hide mapped fields

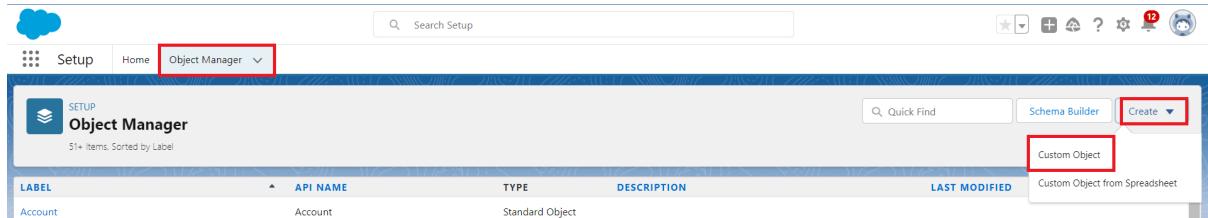
IMPORT FILE FIELD NAME	SALESFORCE FIELD NAME	SALESFORCE FIELD TYPE	ADD TO LAYOUTS	FIELD PREVIEW
Invoice ID	Invoice ID	Text	<input checked="" type="checkbox"/>	
Total Price	Total Price	Integer	<input checked="" type="checkbox"/>	
Quantity	Quantity	Integer	<input checked="" type="checkbox"/>	
Unit Price	Unit Price	Integer	<input checked="" type="checkbox"/>	
Purchase Date	Purchase Date	Date	<input checked="" type="checkbox"/>	

## Activity 3 : Create Automobile Object

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

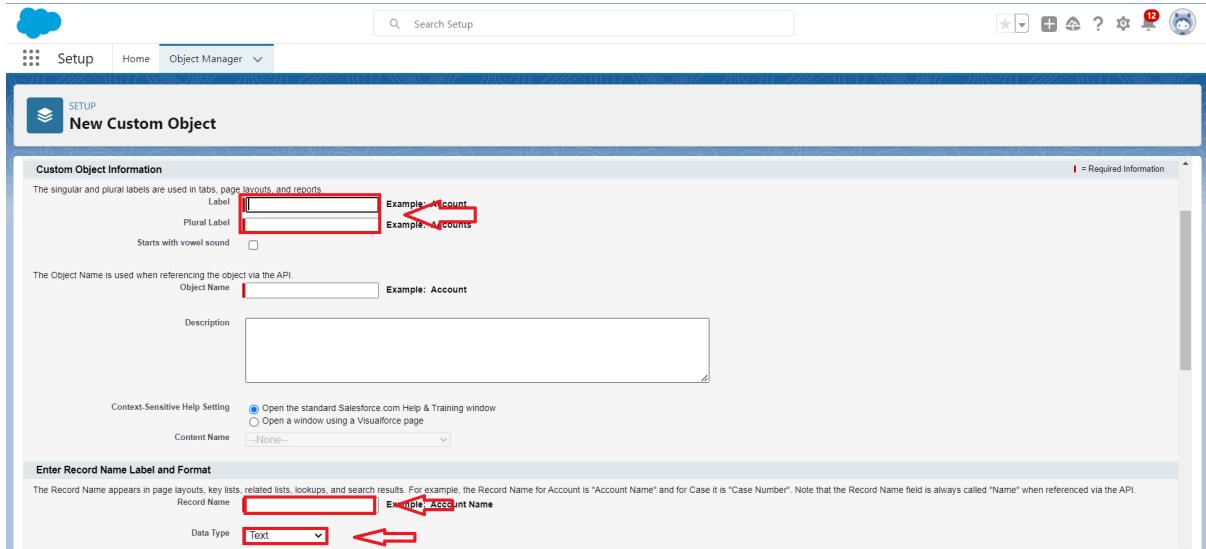
To create an object:

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



The screenshot shows the Salesforce Setup interface. At the top, there's a navigation bar with 'Setup', 'Home', and 'Object Manager'. Below it is a search bar labeled 'Search Setup'. In the main content area, there's a 'Object Manager' section with a blue header. It shows a table with columns: LABEL, API NAME, TYPE, DESCRIPTION, and LAST MODIFIED. There's one row visible for 'Account'. On the far right of the table, there's a 'Create' button. A red box highlights both the 'Object Manager' tab and the 'Create' button.

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



The screenshot shows the 'New Custom Object' page. At the top, there's a 'Custom Object Information' section with fields for 'Label' (containing 'Opportunity Automobile') and 'Plural Label' (containing 'Opportunity Automobiles'). Below this is a 'Description' text area. Further down, there's a 'Context-Sensitive Help Setting' section with two radio buttons: 'Open the standard Salesforce.com Help & Training window' (selected) and 'Open a window using a Visualforce page'. There's also a 'Content Name' dropdown set to 'None'. At the bottom, there's a 'Enter Record Name Label and Format' section. It shows a 'Record Name' field containing 'Opportunity Automobile Id' with an example 'Opportunity Account Name'. Below it is a 'Data Type' dropdown set to 'Text'. Arrows point from the text examples back to their respective input fields.

- 3) Enter Record Name Label and Format

- Record Name → Opportunity Automobile Id
- Data Type → Auto Number
- Display Format → OA-{0000}
- Starting Number → 1

**Custom Object Information**

The singular and plural labels are used in tabs, page layouts, and reports.  
Be careful when changing the name or label as it may affect existing integrations and merge templates.

Label	<input type="text" value="Opportunity Automobile"/>	Example: Account
Plural Label	<input type="text" value="Opportunity Automobiles"/>	Example: Accounts
Starts with vowel sound	<input type="checkbox"/>	

The Object Name is used when referencing the object via the API.

Object Name	<input type="text" value="Opportunity_Automobile"/>	Example: Account
-------------	---	------------------

Description

Context-Sensitive Help Setting

<input checked="" type="radio"/> Open the standard Salesforce.com Help & Training window
<input type="radio"/> Open a window using a Visualforce page

Content Name

Enter Record Name Label and Format

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". No

Record Name	<input type="text" value="Opportunity Automobile Id"/>	Example: Account Name
Data Type	<input type="text" value="Auto Number"/>	
Display Format	<input type="text" value="OA-{0000}"/>	Example: A-{0000} <a href="#">What Is This?</a>
Starting Number	<input type="text" value="1"/>	

2. Click on Allow reports.
3. Allow search
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)

### Custom Tabs

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external links. Lightning Component tabs allow you to add Lightning components to the navigation bar. This section will guide you to add Lightning Pages to Lightning Experience and the mobile app.

The screenshot shows two sections of the Salesforce Setup interface:

- Custom Object Tabs:** A header with "Custom Object Tabs" and "New" (which is highlighted with a red box) and "What Is This?". Below it, a message says "No Custom Object Tabs have been defined".
- Web Tabs:** A header with "Web Tabs" and "New" (highlighted with a red box) and "What Is This?". Below it, a message says "No Web Tabs have been defined".

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.

The screenshot shows the 'Custom Object Tab Definition Detail' page for an object named 'Opportunity Automobile'. The page includes fields for Tab Label (Opportunity Automobile), Object (Opportunity Automobile), Description (empty), and Created By (Mohammad\_Sameer, 27/11/2023, 2:48 pm). It also shows Tab Style (Bell), Splash Page Custom Link, and Modified By (Mohammad\_Sameer, 27/11/2023, 2:48 pm). Buttons for Edit and Delete are at the top right, and a Help for this link is at the top right.

**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.

The screenshot shows the Salesforce App Manager interface. At the top, there are tabs for 'Setup', 'Home', and 'Object Manager'. Below these, a search bar says 'Search Setup'. On the left, there's a sidebar with 'Q app manager' and 'App Manager' selected. In the center, it says 'Lightning Experience App Manager' and 'Clone (App,Beta)'. A red arrow points to the 'New Lightning App' button at the top right. Below this, there's a note about cloning existing apps and enabling app cloning. The main area is a table listing 35 apps, with columns for App Name, Developer Name, Description, Last Modified, App Type, and more. A red arrow also points to the table header.

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.

The screenshot shows the 'New Lightning App' configuration page. It has two main sections: 'App Details & Branding' and 'App Launcher Preview'. In the 'App Details & Branding' section, there are fields for 'App Name' (with a red arrow pointing to it), 'Developer Name' (auto-populated), 'Description', 'Image' (with an 'Upload' button), 'Primary Color Hex Value' (#0070D2), and 'Org Theme Options' (unchecked). At the bottom, there's a 'Next' button with a red arrow pointing to it.

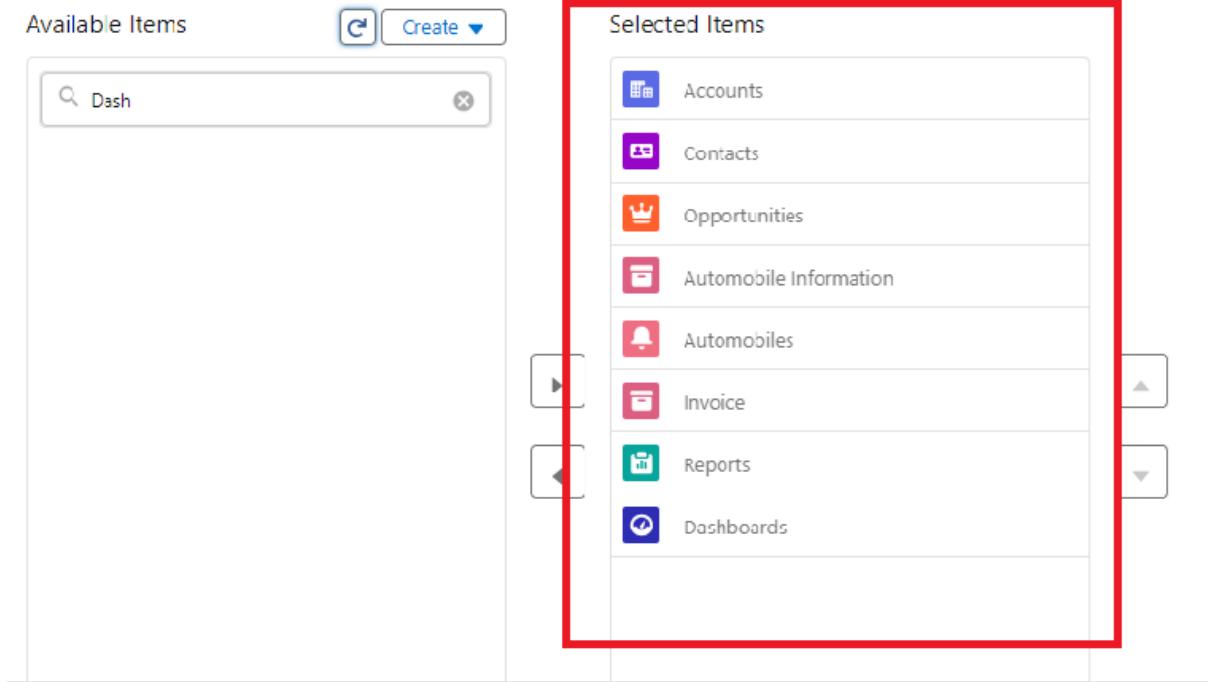
4. To Add Navigation Items:

New Lightning App

---

## Navigation Items

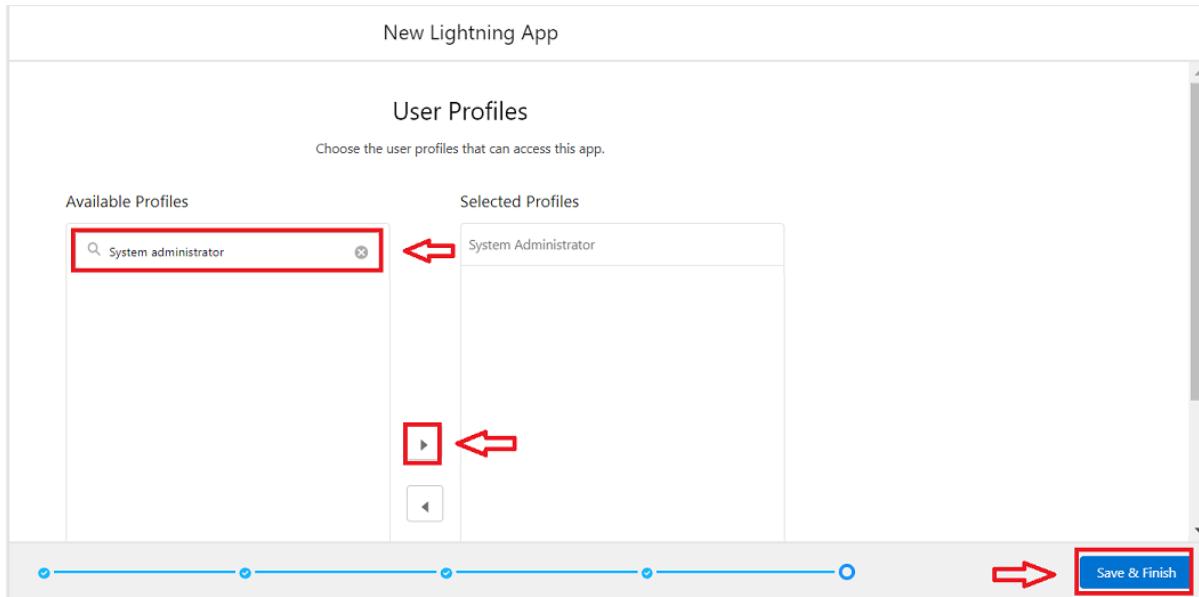
By default, users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.



Item	Description
Accounts	
Contacts	
Opportunities	
Automobile Information	
Automobiles	
Invoice	
Reports	
Dashboards	

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.

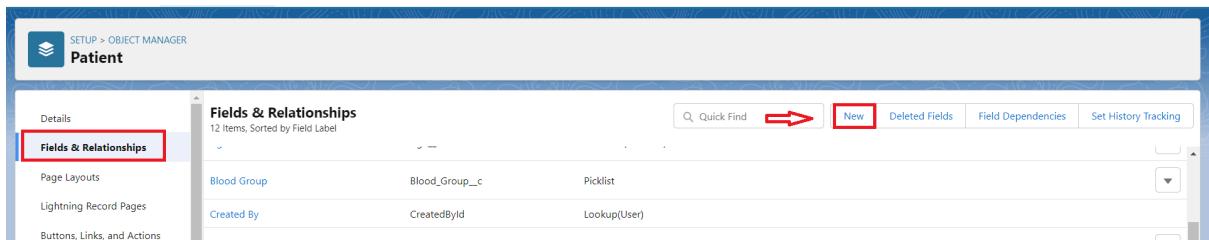
### **Activity 1: Creating Opportunity Master Detail Relationship Field in Opportunity AutoMobile Object**

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”.

SETUP > OBJECT MANAGER

# Opportunity Automobile

**Details**

**Fields & Relationships**

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Opportunity Automobile  
New Custom Field

**Step 1. Choose the field type**

Specify the type of information that the custom field will contain.

**Data type**

None Selected Select one of the data types below.

Auto Number A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.

Formula A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.

Roll-Up Summary A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.

Lookup Relationship Creates a relationship that links this object to another object. The relationship field allows users to click on a lookup icon to select a value from a pop-up list. The object selected must have a master-detail relationship with this object.

Master-Detail Relationship Creates a special type of parent-child relationship between this object (the child, or "detail") and another object (the parent, or "master") where:

- The relationship field is required on all detail records.
- The ownership and sharing of a detail record are determined by the master record.
- When a user deletes the master record, all detail records are deleted.
- You can create roll-up summary fields on the master record to summarize the detail records.

The relationship field allows users to click on a lookup icon to select a value from a pop-up list. The master object is the source of the values in the list.

External Lookup Relationship Creates a relationship that links this object to an external object whose data is stored outside the Salesforce org.

#### 4. Click on Next

Opportunity Automobile  
New Relationship

Help for this Page ?

Step 2. Choose the related object Step 2 of 6

Select the other object to which this object is related.

Related To

Previous Next Cancel

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.

Opportunity Automobile  
New Relationship

Step 3. Enter the label and name for the lookup field

Step 3 of 6

Field Label: Opportunity [ ]

Field Name: Opportunity [ ]

Description: [ ]

Help Text: [ ]

Child Relationship Name: Opportunity\_Automobiles [ ]

Sharing Setting: Select the minimum access level required on the Master record to create, edit, or delete related Detail records.

Read Only: Allows users with at least Read access to the Master record to create, edit, or delete related Detail records.

Read/Write: Allows users with at least Read/Write access to the Master record to create, edit, or delete related Detail records.

Allow reparenting:  Child records can be reparented to other parent records after they are created

Auto add to custom report type:  Add this field to existing custom report types that contain this entity [ ]

**Lookup Filter**

Optional: create a filter to limit the records available to users in the lookup field. [Tell me more!](#)

▶ [Show Filter Settings](#)

Previous Next Cancel

## Activity 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.

The screenshot shows the Salesforce Object Manager interface. In the search bar at the top right, 'Opportunity Automobile' is typed. Below the search bar, a table lists objects. The first row, 'Opportunity Automobile', has its 'Label' ('Opportunity Automobile'), 'API Name' ('Opportunity\_Automobile\_\_c'), and 'Type' ('Custom Object') highlighted with a red box. The 'Last Modified' field shows '27/11/2023'. A 'Create' button is visible in the top right corner of the table header.

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

The screenshot shows the 'Fields & Relationships' page for the 'Patient' object. On the left, there are navigation links: 'Details', 'Fields & Relationships' (which is selected and highlighted with a red box), 'Page Layouts', 'Lightning Record Pages', and 'Buttons, Links, and Actions'. The main area displays two fields: 'Blood Group' (of type 'Picklist') and 'Created By' (of type 'Lookup(User)'). At the top right, there is a 'New' button highlighted with a red box and an arrow pointing to it.

3. Select Data type as “Lookup RelationShip”.

The screenshot shows the 'Data Type' selection screen for the 'Opportunity Automobile' object. On the left, there are navigation links: 'Details', 'Fields & Relationships' (selected and highlighted with a red box), 'Page Layouts', 'Lightning Record Pages', 'Buttons, Links, and Actions', 'Compact Layouts', 'Field Sets', and 'Object Limits'. The main area shows the 'Data Type' section with 'None Selected' as the current choice. Below it, three other options are listed: 'Auto Number', 'Formula', and 'Roll-up Summary'. The 'Lookup Relationship' option is highlighted with a red box. A detailed description of 'Lookup Relationship' is provided on the right side of the screen.

4. Click on Next

The screenshot shows the 'Step 2. Choose the related object' screen. At the top, it says 'Opportunity Automobile' and 'New Relationship'. Below that, it says 'Select the other object to which this object is related.' A dropdown menu labeled 'Related To' contains the value 'Automobile Information', which is highlighted with a red box. At the bottom right, there are 'Previous', 'Next', and 'Cancel' buttons, with 'Next' highlighted with a red box.

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

Opportunity Automobile  
New Relationship

Help for this Page 

Step 3. Enter the label and name for the lookup field Step 3 of 6

Field Label  

Field Name  

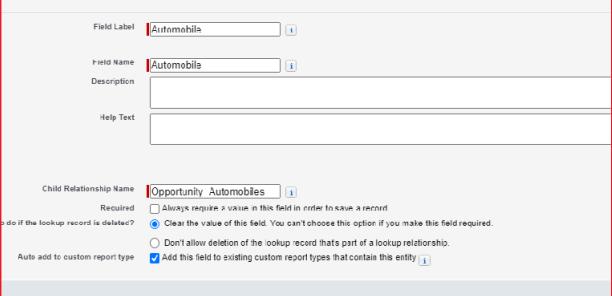
Description

Help Text

Child Relationship Name  

Required  Always require a value in this field in order to save a record  
 Clear the value of this field. You can't choose this option if you make this field required.  
 Don't allow deletion of the lookup record that's part of a lookup relationship.

Auto add to custom report type  Add this field to existing custom report types that contain this entity 

Lookup Filter 

Optional, create a filter to limit the records available to users in the lookup field. [Tell me more!](#)  
[Show Filter Settings](#)

Previous  Next  Cancel

## 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.

Opportunity Automobile

Formula   
A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.

Roll-Up Summary   
A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.

Lookup Relationship   
Creates a relationship that links this object to another object. The relationship field allows users to click on a lookup icon to select a value from a popup list.

Master-Detail Relationship   
Creates a special type of parent-child relationship between this object (the child, or "detail") and another object (the parent, or "master") where:  

- The relationship field is required on all detail records.
- The ownership and sharing of a detail record are determined by the master record.
- When a user deletes the master record, all detail records are deleted.
- You can create roll-up summary fields on the master record to summarize the detail records.

The relationship field allows users to click on a lookup icon to select a value from a popup list. The master object is the source of the values in the list.

External Lookup Relationship   
Creates a relationship that links this object to an external object whose data is stored outside the Salesforce org.

Checkbox   
Allows users to select a True (checked) or False (unchecked) value.

Currency   
Allows users to enter a dollar or other currency amount and automatically formats the field as a currency amount. This can be useful if you export data to Excel.

Date   
Allows users to enter a date or pick a date from a popup calendar.

Date/Time   
Allows users to enter a date and time, or pick a date from a popup calendar. When users click a date in the pop-up, that date and the current time are entered.

Email   
Allows users to enter an email address, which is validated to ensure proper format. If this field is specified for a contact or lead, users can choose the add mass emails.

Geolocation   
Allows users to define locations, includes latitude and longitude components, and can be used to calculate distance.

Number   
 Allows users to enter any number. Leading zeros are removed.

Percent   
Allows users to enter a percentage number, for example, 10% and automatically adds the percent sign to the number.

Phone   
Allows users to enter any phone number. Automatically formats it as a phone number.

Picklist   
Allows users to select a value from a list you define.

Picklist (Multi-Select)   
Allows users to select multiple values from a list you define.

a. Field Label → Quantity  
b. Field Name→ Quantity

Opportunity Automobile | Sales | Home | Salesforce | Object Manager | Salesforce | thesmartbridge-21e-dev-ed.lightning.force.com/lightning/setup/ObjectManager/0115j000002rltN/FieldsAndRelationships/new

Setup | Home | Object Manager

SETUP > OBJECT MANAGER  
Opportunity Automobile

Step 2. Enter the details Step 2 of 4

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Field Label  Field Name

Please enter the length of the number and the number of decimal places. For example, a number with a length of 8 and 2 decimal places can accept values up to "12345678.90".

Length  Number of digits to the left of the decimal point

Decimal Places  Number of digits to the right of the decimal point

Description

Help Text

Previous Next Cancel

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.

Opportunity Automobile  
New Custom Field

Step 2. Choose output type Step 2 of 5

Field Label  Field Name

Auto add to custom report type  Add this field to existing custom report types that contain this entry

Formula Return Type

None Selected

Currency

Checkbox

Date

Date/Time

Number

Percent

Text

Time

Options

Select one of the data types below.

Calculate a boolean value.  
Example: `(TODAY() > CloseDate)`

Calculate a dollar or other currency amount and automatically format the field as a currency amount.  
Example: `(Gross Margin * Amount) / Cost_LvL`

Calculate a date, for example, by adding or subtracting days to other dates.  
Example: `Reminder Date = CloseDate - 7`

Calculate a datetime, for example, by adding a number of hours or days to another datetime.  
Example: `LeaveTime = If(7W) + 1`

Calculate a number value.  
Example: `Fahrenheit = 1.8 * Celsius + 32`

Calculate a percent and automatically add the percent sign to the number.  
Example: `Discount = (Amount - Discounted_Amount) / Amount`

Create a text string, for example, by concatenating other text fields.  
Example: `Full Name = LastName & ", " & FirstName`

Calculate a time, for example, by adding a number of hours to another time.  
Example: `Next = TIMEVALUE(NOW()) + 1`

Decimal Places  Example: 099.00

Previous Next Cancel

4. Under Advanced Formula write down the formula : **Automobile\_\_r.Price\_\_c**

The screenshot shows the Salesforce formula editor. At the top, there are tabs for "Simple Formula" and "Advanced Formula", with "Advanced Formula" selected. Below the tabs are buttons for "Insert Field" and "Insert Operator". The main input field contains the formula "Automobile\_\_r.Price\_\_c", which is also enclosed in a red rectangular box. At the bottom of the editor, there is a status message: "Check Syntax" followed by "No syntax errors in merge fields or functions. (Compiled size: 31 characters)".

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.
6. Now click on “Fields & Relationships” → New.
7. Select Data type as “Formula” and click Next.
8. 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.

Opportunity Automation  
New Custom Field

Step 2 of 5

**Step 2. Choose output type**

Field Label: **Total Price**

Field Name: **Total\_Price**

Auto add to custom report type  Add this field to existing custom report types that contain this entity

**Formula Return Type**

None Selected

Currency

Date

Date/Time

Number

Percent

Text

Time

Selected one of the data types below.

Calculate a boolean value.  
Example: `(ClosedDate > CloseDate)`

Calculate a date or time or currency amount and automatically format the field as a currency amount.  
Example: `(Gross Margin - Amount) / Cost__c`

Calculate a date, for example, by adding or subtracting days to other dates.  
Example: `(Reminder Date + CloseDate - 7)`

Calculate a datetime, for example, by adding a number of hours or days to another datetime.  
Example: `(Next = NOW() + 1)`

Calculate a number value.  
Example: `Fahrenheit * 1.8 + Celsius * 32`

Calculate a percent and automatically add the percent sign to the number.  
Example: `Discount / (Amount - Discounted_Amount__c) / Amount`

Create a text string, for example, by concatenating other text fields.  
Example: `Full Name & Last Name & " " & First Name`

Calculate a time, for example, by adding a number of hours to another time.  
Example: `(Next = TIMEVALUE(NOW()) + 1)`

Options

9. Under Advanced Formula write down the formula : **Unit\_Price\_\_c \* Quantity\_\_c**

Example: `Gross Margin - Amount - Cost__c` [More Examples...](#)

Simple Formula  Advanced Formula

Insert Field  Insert Operator

`Total_Price (Currency)`

`Unit_Price__c * Quantity__c`

No syntax errors in merge fields or functions. (Compiled size: 75 characters)

10. 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.

Fields & Relationships				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User/Group)		✓
Purchase Date	Purchase_Date__c	Date		
Quantity	Quantity__c	Number(18, 0)		
Total Price	Total_Price__c	Number(18, 0)		
Unit Price	Unit_Price__c	Number(18, 0)		
Invoice ID	Name	Text(80)		✓

SETUP > OBJECT MANAGER  
**Invoice**

Details	Field
Fields & Relationships	<b>Invoice ID</b>
Page Layouts	The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" referenced via the API.
Lightning Record Pages	
Buttons, Links, and Actions	
Compact Layouts	
Field Sets	
Object Limits	
Record Types	
Related Lookup Filters	
Search Layouts	

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.

s.no	Object name	Fields

1	Invoice	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Field Name</td><td style="padding: 5px;">Opportunity</td></tr> <tr> <td style="padding: 5px;">Data type</td><td style="padding: 5px;">Master Detail relationship Object : Opportunity</td></tr> </table>	Field Name	Opportunity	Data type	Master Detail relationship Object : Opportunity
Field Name	Opportunity					
Data type	Master Detail relationship Object : Opportunity					

## **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**

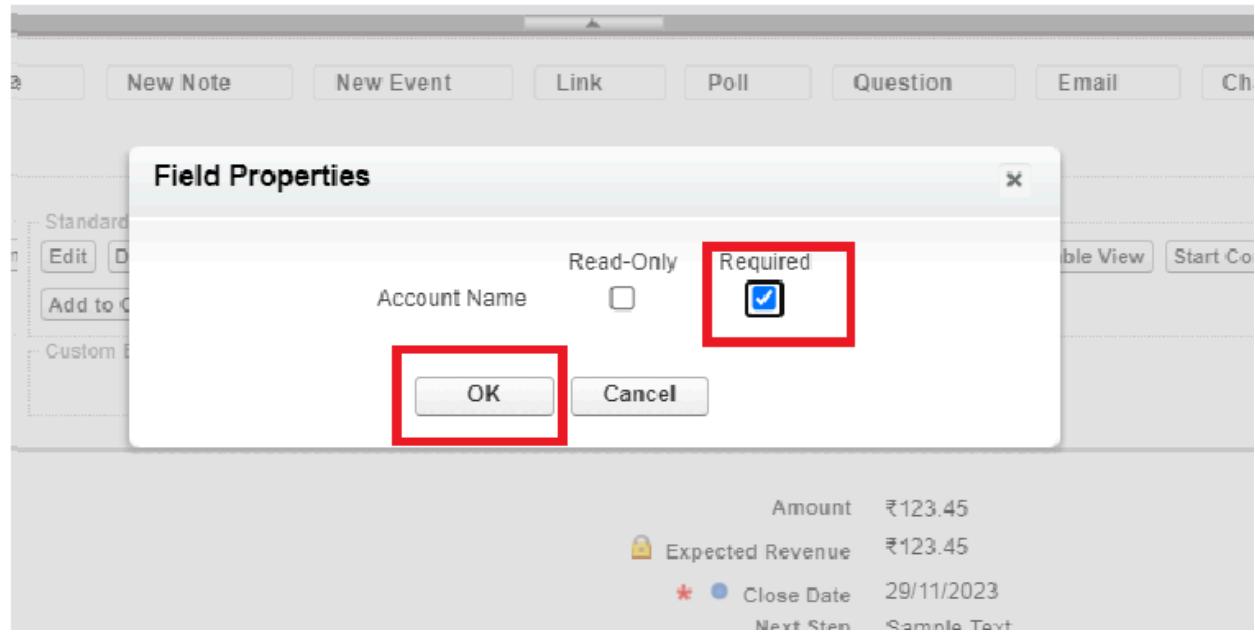
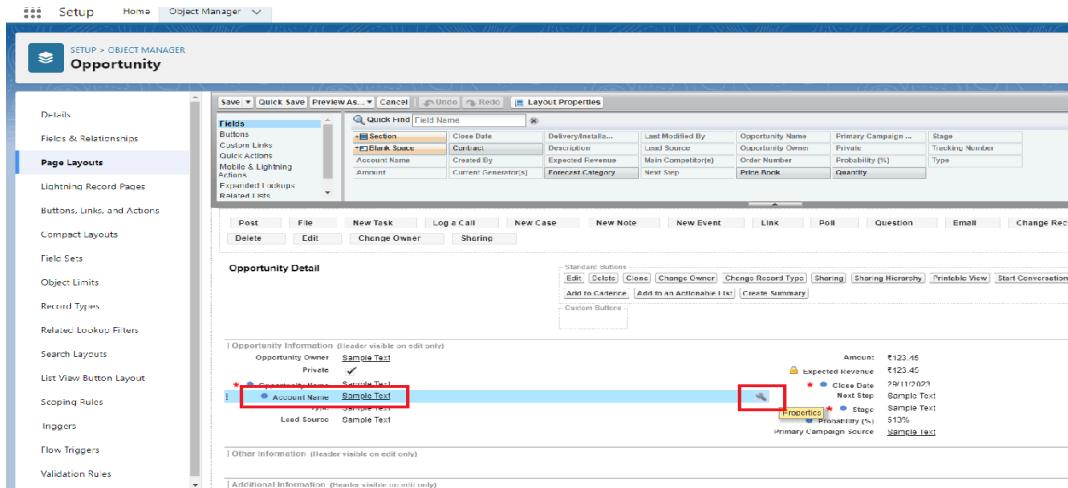
Step 1: Go to Setup → Click on Object Manager → On the search bar, select Opportunity Layout. You can notice Page Layouts on the left panel

Step 2: Click on Page Layouts, Click on ‘Opportunity Layout’.

The screenshot shows the Salesforce Object Manager interface for the Opportunity object. The left sidebar has 'Page Layouts' selected. The main area displays a table of page layouts:

PAGE LAYOUT NAME	CREATED BY	MODIFIED BY
Opportunity (Marketing) Layout	Mohammad Samer 22/11/2023 2:19 pm	Mohammad Samer 28/11/2023 9:15 am
Opportunity (Sales) Layout	Mohammad Samer 22/11/2023 2:19 pm	Mohammad Samer 28/11/2023 9:45 am
Opportunity (Support) Layout	Mohammad Samer 22/11/2023 2:19 pm	Mohammad Samer 28/11/2023 9:47 am
Opportunity Layout	Mohammad Samer 22/11/2023 2:19 pm	Mohammad Samer 28/11/2023 9:15 am

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’.

The image consists of two screenshots of the Salesforce Object Manager interface. The top screenshot shows the 'Page Layouts' list for the 'Automobile Information' object. The bottom screenshot shows the edit screen for the 'Automobile Information Layout'.

**Screenshot 1: Page Layouts List**

PAGE LAYOUT NAME	CREATED BY	MODIFIED BY
Automobile Information Layout	Mohammad Sameer, 27/11/2023, 12:49 pm	Mohammad Sameer, 27/11/2023, 10:32 pm

**Screenshot 2: Page Layout Editor**

The editor shows the 'Fields' section with various fields listed:

- Buttons
- Quick Actions
- Mobile & Lightning Actions
- Exposed Lookups
- Related Lists
- Report Charts

Below the fields, there are sections for 'Highlights Panel', 'Quick Actions in the Salesforce Classic Publisher', and 'Salesforce Mobile and Lightning Experience Actions'. At the bottom, there is an 'Automobile Information Detail' section with standard buttons: Edit, Delete, Clone, Change Owner, Change Record Type, Printable View, Sharing, Sharing Hierarchy, Submit for Approval, and Custom Buttons.

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 Then 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;
}

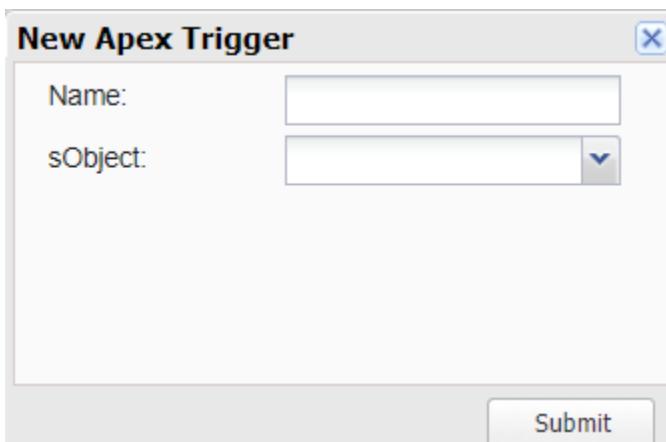
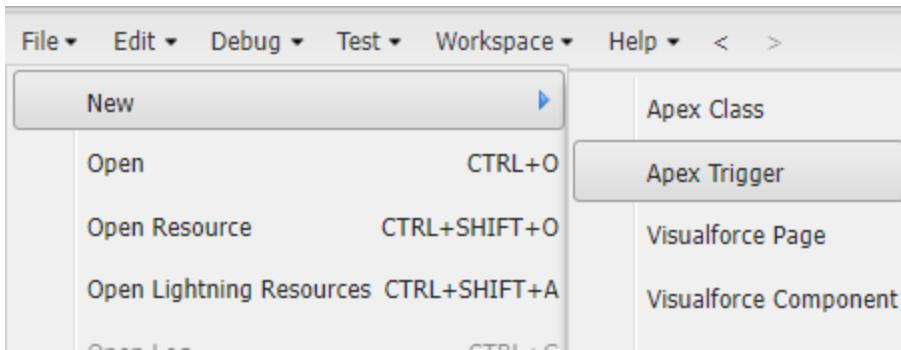
}
}

```

**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){  
    //block of code  
}
```

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.

```

1 trigger OpportunityTrigger on Opportunity (before update, After Update) {
2     if(trigger.isbefore && trigger.isUpdate){
3         OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
4     }
5 }

```

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 ”.

```

1 public class OpportunityAutomobileHandler {
2     public static void quantityErrorOnAutomobileInformation(List<Opportunity_Automobile__c> lstOpportunityAutomobile){
3         Set<Id> AutomobileIds = new Set<Id>();
4         Map<Id, Automobile_Information__c> lstAutomobileInformation = new Map<Id, Automobile_Information__c>();
5         For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
6             if(OppAutomobile.Automobile__c != null){
7                 AutomobileIds.add(OppAutomobile.Automobile__c);
8             }
9         }
10         Map<Id, Automobile_Information__c> lstAutomobileInformation = new Map<Id, Automobile_Information__c>([SELECT Id, CreatedById, Quantity__c, Price__c
11                                         FROM Automobile_Information__c WHERE Id IN: AutomobileIds]);
12         For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
13             If(OppAutomobile.Automobile__c == lstAutomobileInformation.get(OppAutomobile.Automobile__c).Id && lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c < OppAutomobile.Quantity__c){
14                 OppAutomobile.addError('the Number of Automobile u want are not Available !! the Automobile are Available Count is ' + lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c );
15             }
16         }
17     }
}

```

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);
            }
        }
    }
}

```

```

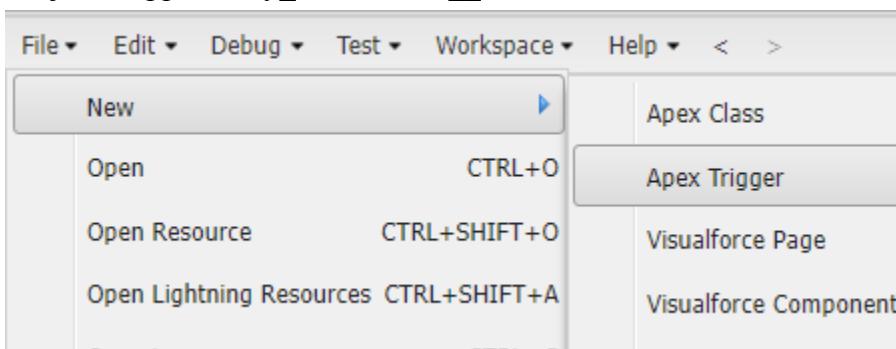
Map<Id, Automobile_Information__c> lstAutomobileInformation = new
map<Id, Automobile_Information__c>([SELECT Id, CreatedById, Quantity__c, Price__c FROM
Automobile_Information__c WHERE Id IN: AutomobileIds]);
For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
    If(OppAutomobile.Automobile__c ==
lstAutomobileInformation.get(OppAutomobile.Automobile__c).Id &&
lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c <
OppAutomobile.Quantity__c){
        OppAutomobileaddError('the Number of Automobile u want are not Available !! the
Automobile are Available Count is ' +
lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c );
    }
}
}

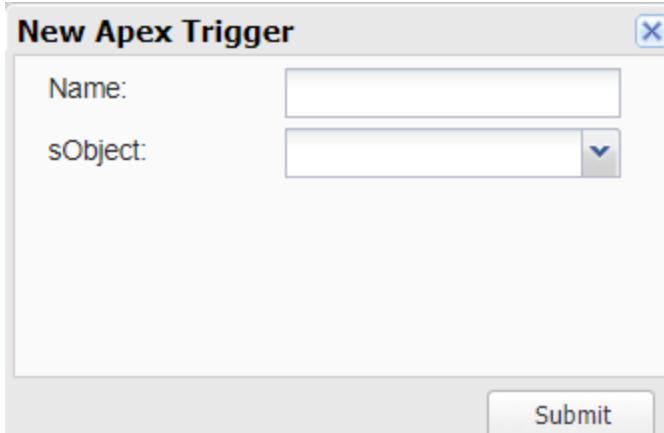
```

### **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

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

Code Coverage: None API Version: 59

```

1 trigger OpportunityAutoMobileTrigger on Opportunity_Automobile__c (before insert, before update) {
2     if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
3         OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(trigger.new);
4     }
5 }
```

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”.

```

1 public class InvoiceCreation {
2     public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity> lstOpportunity, Map<Id,Opportunity>OldMapOpportunity){
3         set<Id> oppIds = new Set<Id>();
4         For(Opportunity opp : lstOpportunity){
5             if(Opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName != opp.StageName){
6                 oppIds.add(opp.Id);
7             }
8         }
9         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];
10        List<Invoice__c> lstInvoice = new List<Invoice__c>();
11        For(Opportunity_Automobile__c oppAuto : lstOpportunityAutomobile){
12            Invoice__c i = new Invoice__c();
13            i.Quantity__c = oppAuto.Quantity__c;
14            i.Unit_Price__c = oppAuto.Unit_Price__c;
15            i.Total_Price__c = oppAuto.Total_Price__c;
16            i.Purchase_Date__c = date.today();
17            i.Opportunity__c = oppAuto.Opportunity__c;
18            lstInvoice.add(i);
19        }
20        if(!lstInvoice.isEmpty()){
21            insert lstInvoice;
22        }
23    }
24 }

```

**Code:**

```

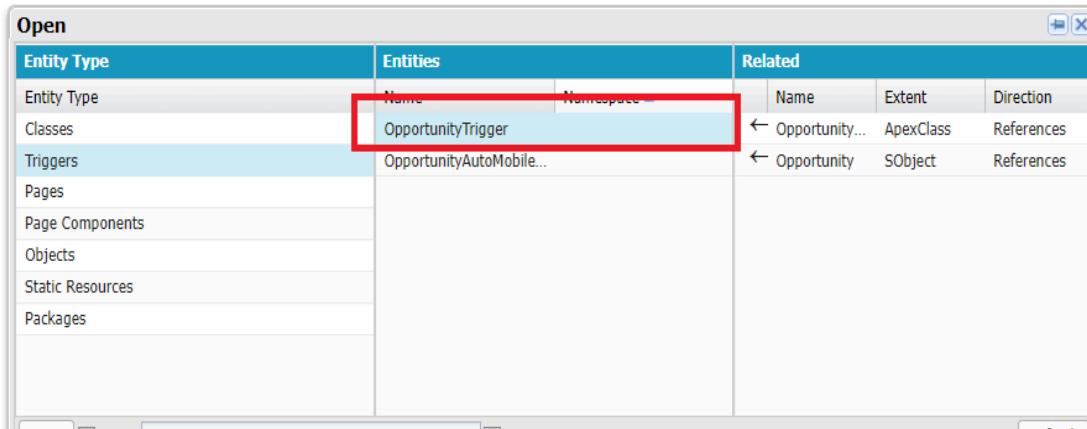
public class InvoiceCreation {
    public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity>
lstOpportunity, Map<Id,Opportunity>OldMapOpportunity){
        set<Id> oppIds = new Set<Id>();
        For(Opportunity opp : lstOpportunity){
            if(Opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName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



The screenshot shows the code editor with the following trigger definition:

```
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);
    }
}
```

## 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.');//  
                }  
            }  
        }  
    }  
}
```

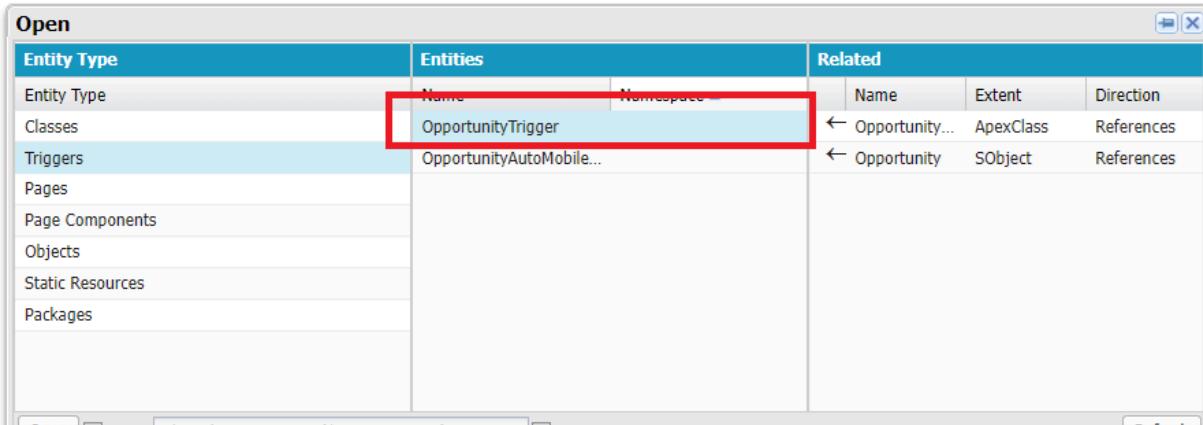
Trigger:

```
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
1 trigger OpportunityTrigger on Opportunity (before update, After Update) {
2     if(trigger.isbefore && trigger.isUpdate){
3         OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
4         ContactRoleCheck.CheckcontactRoleonOpportunity(trigger.new, trigger.oldMap);
5     }
6     IF(trigger.isafter && trigger.isupdate){
7         InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);
8     }
9 }
```

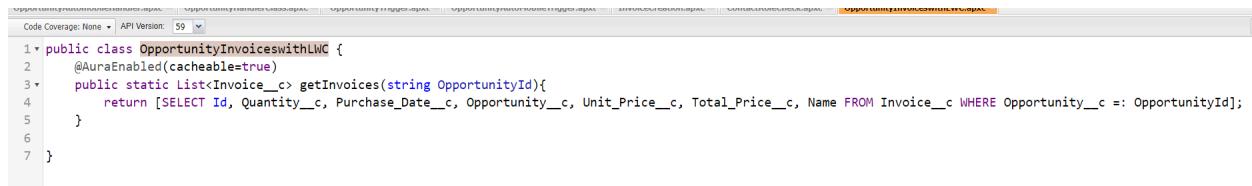
### Trigger:

```
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 Coverage: None | API Version: 59
1 • public class OpportunityInvoiceswithLWC {
2     @AuraEnabled(cacheable=true)
3     public static List<Invoice__c> getInvoices(string OpportunityId){
4         return [SELECT Id, Quantity__c, Purchase_Date__c, Opportunity__c, Unit_Price__c,
5             Total_Price__c, Name FROM Invoice__c WHERE Opportunity__c =: OpportunityId];
6     }
7 }
```

### 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.

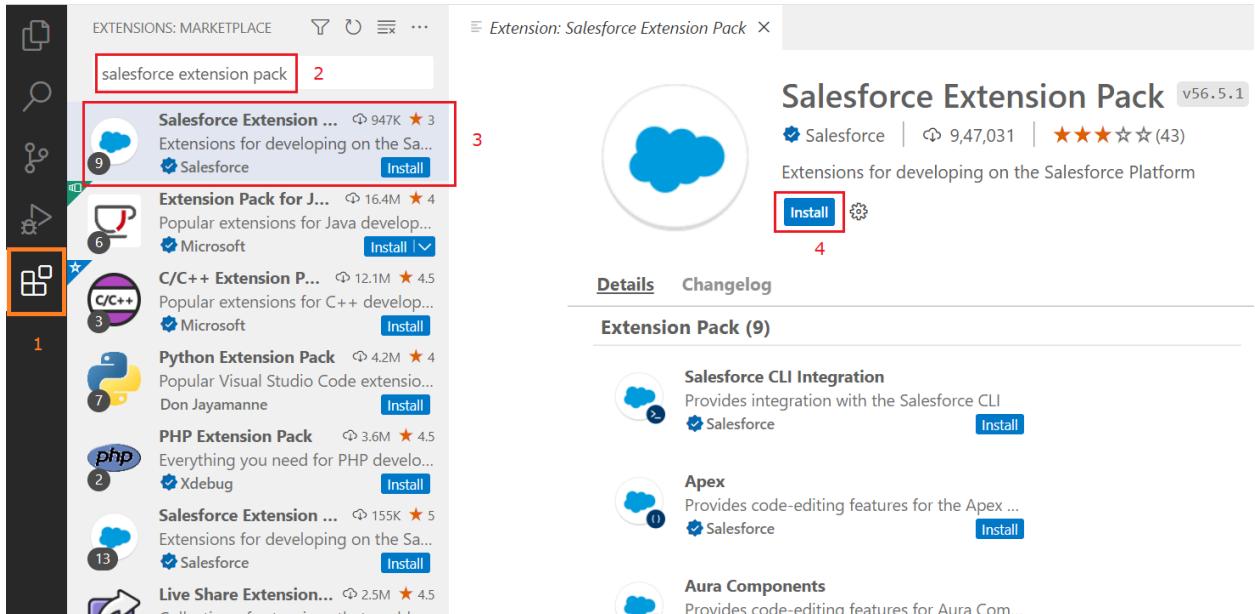
[Download the version of the software](#) that is compatible with your operating system and install it.

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

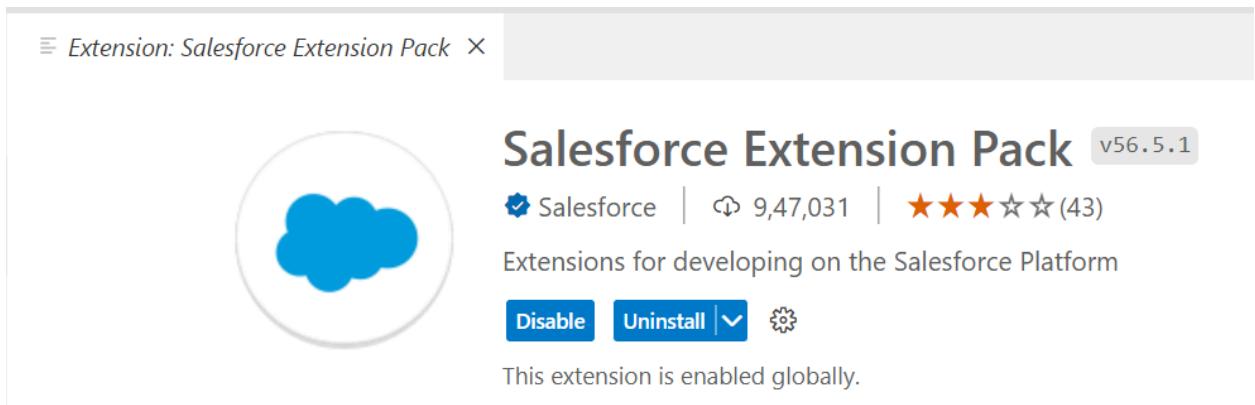
### Activity- 3: Install the Salesforce Extension Pack

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.



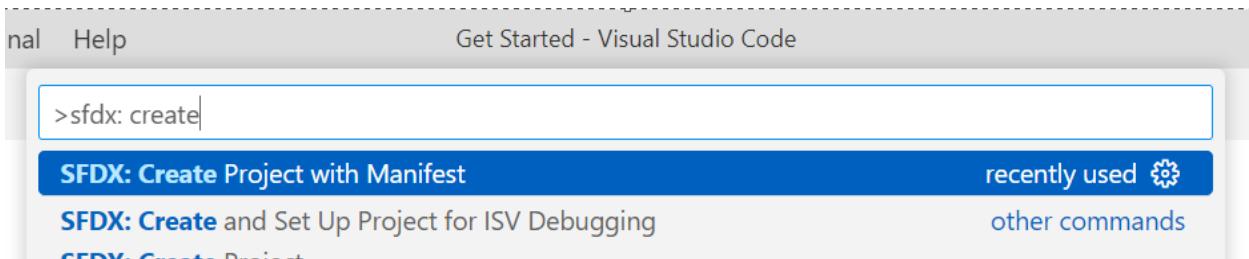
The extension pack is installed successfully



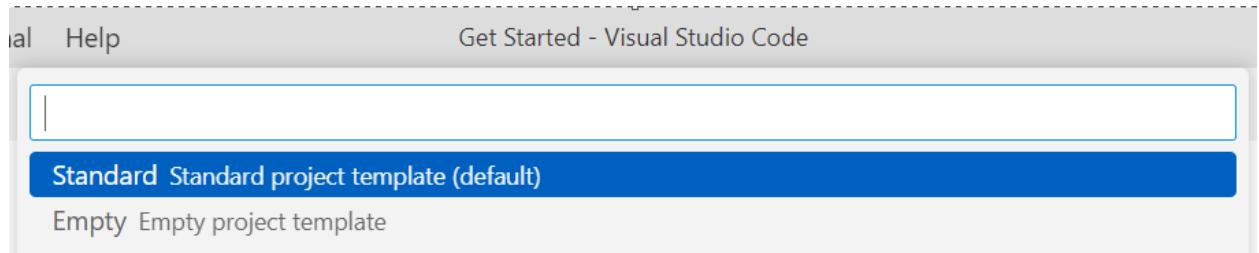
## Install the Salesforce Extension Pack

### Activity- 4 : Create a project in VS Code

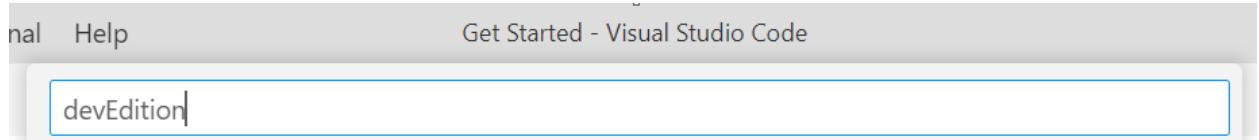
1. Press CTRL + SHIFT + P, type sfdx: create
2. select SFDX: Create Project with Manifest



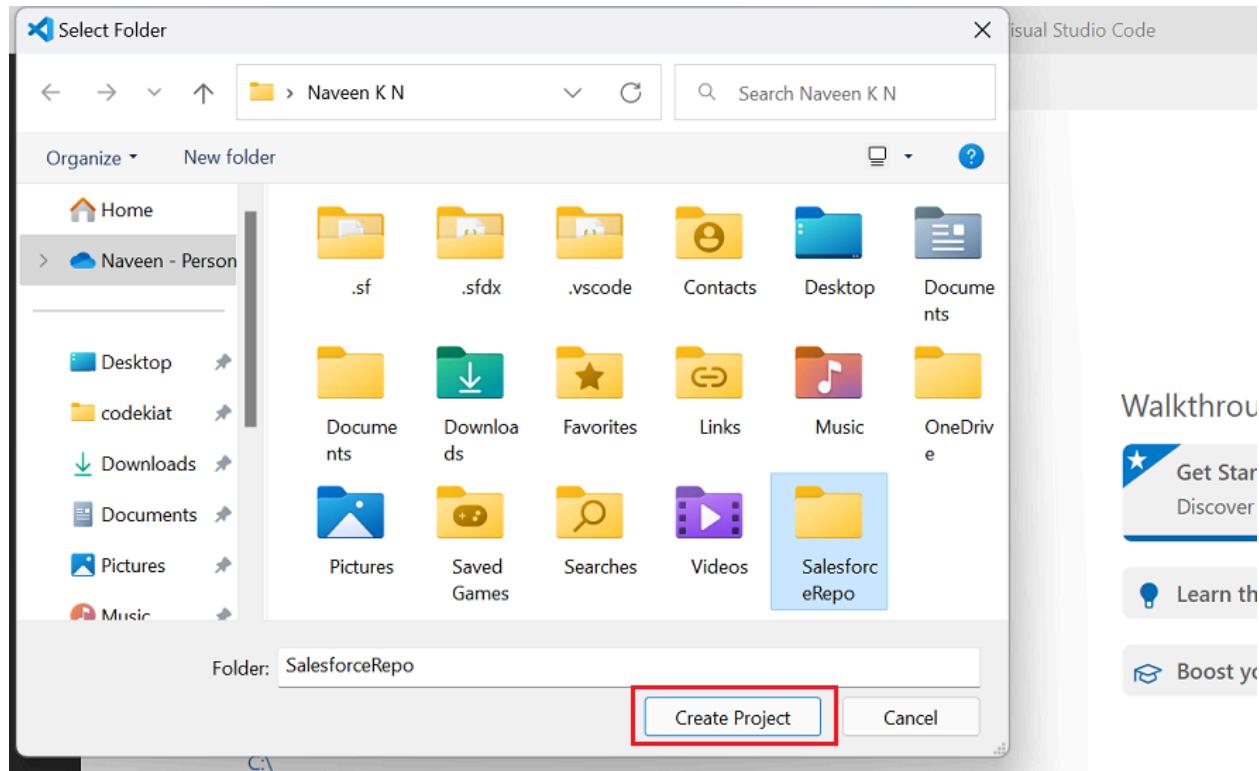
3. Select the Standard project template



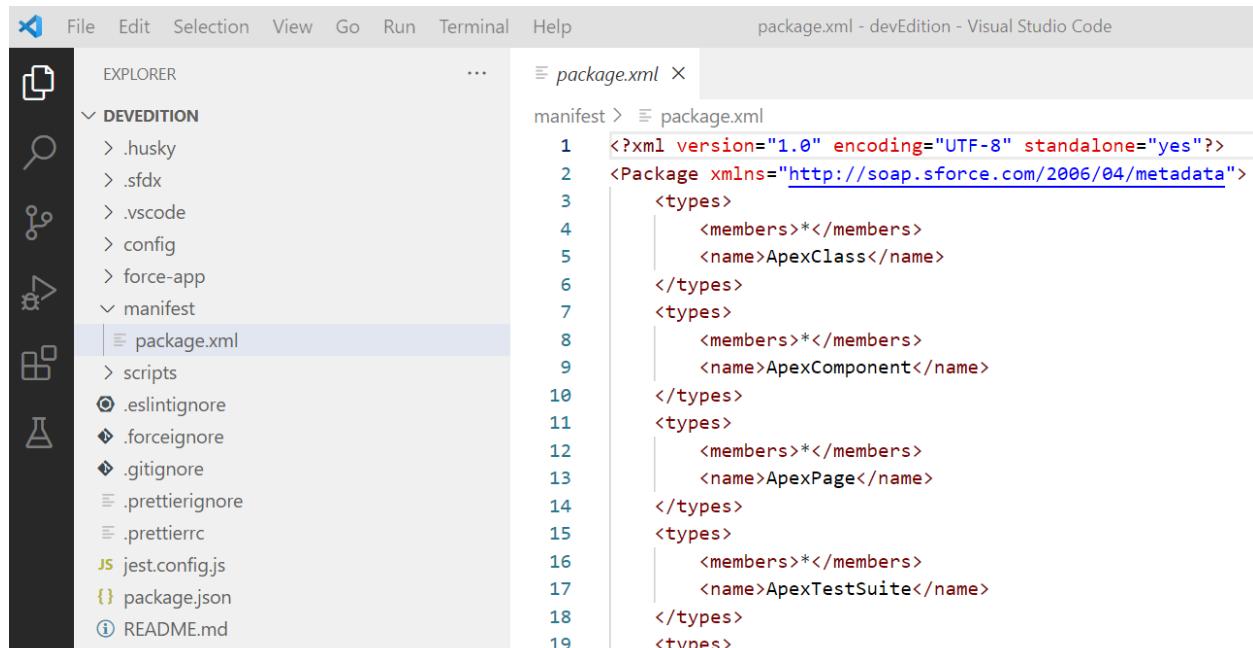
4. Type a project name and Click Enter.



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



## 6. The new project is created with package.xml



The screenshot shows the Visual Studio Code interface. The left sidebar is the Explorer view, showing a project structure for 'DEVEDITION'. The 'package.xml' file is selected and highlighted in blue. The main editor area shows the XML code for a package manifest. The code includes declarations for ApexClass, ApexComponent, ApexPage, and ApexTestSuite.

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

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

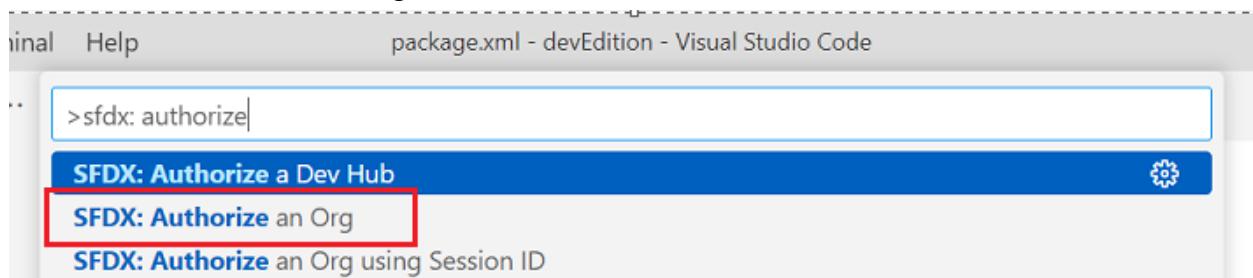
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>LightningComponentBundle</name>
    </types>
    <version>55.0</version>
</Package>
```

### Activity- 5 : Authorize an org

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

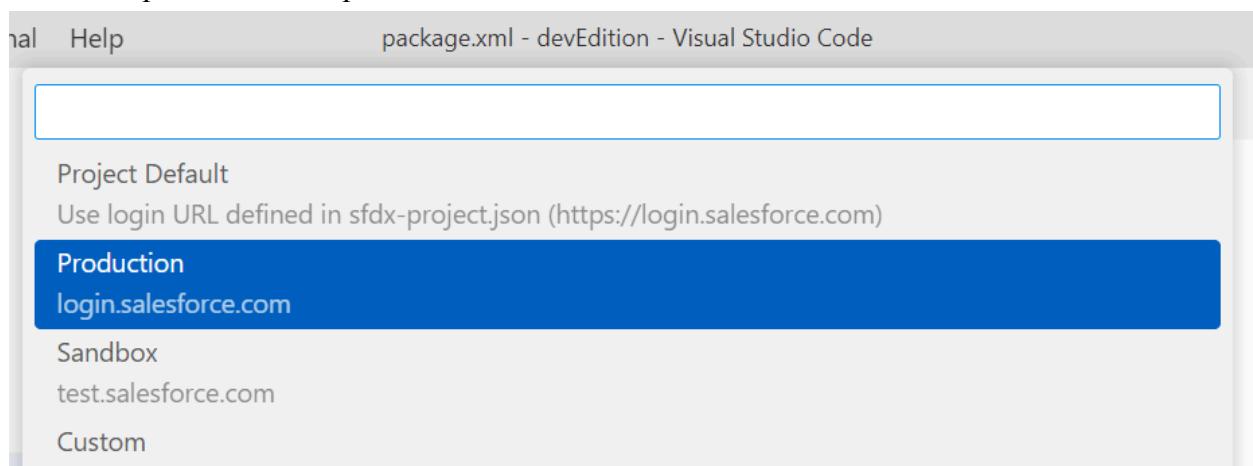
1. Press CTRL + SHIFT + P, type sf: authorize.

2. select SFDX: Authorize an Org from the list



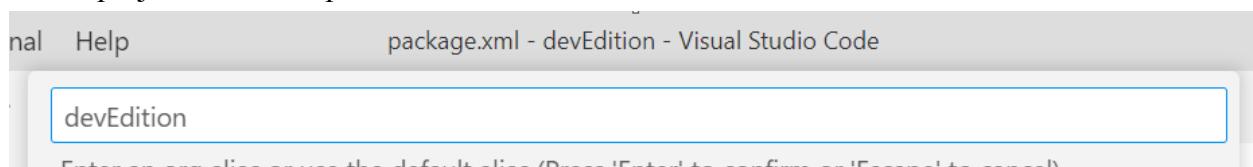
3. Choose your Salesforce instance.

For developer edition and production instances select Production.



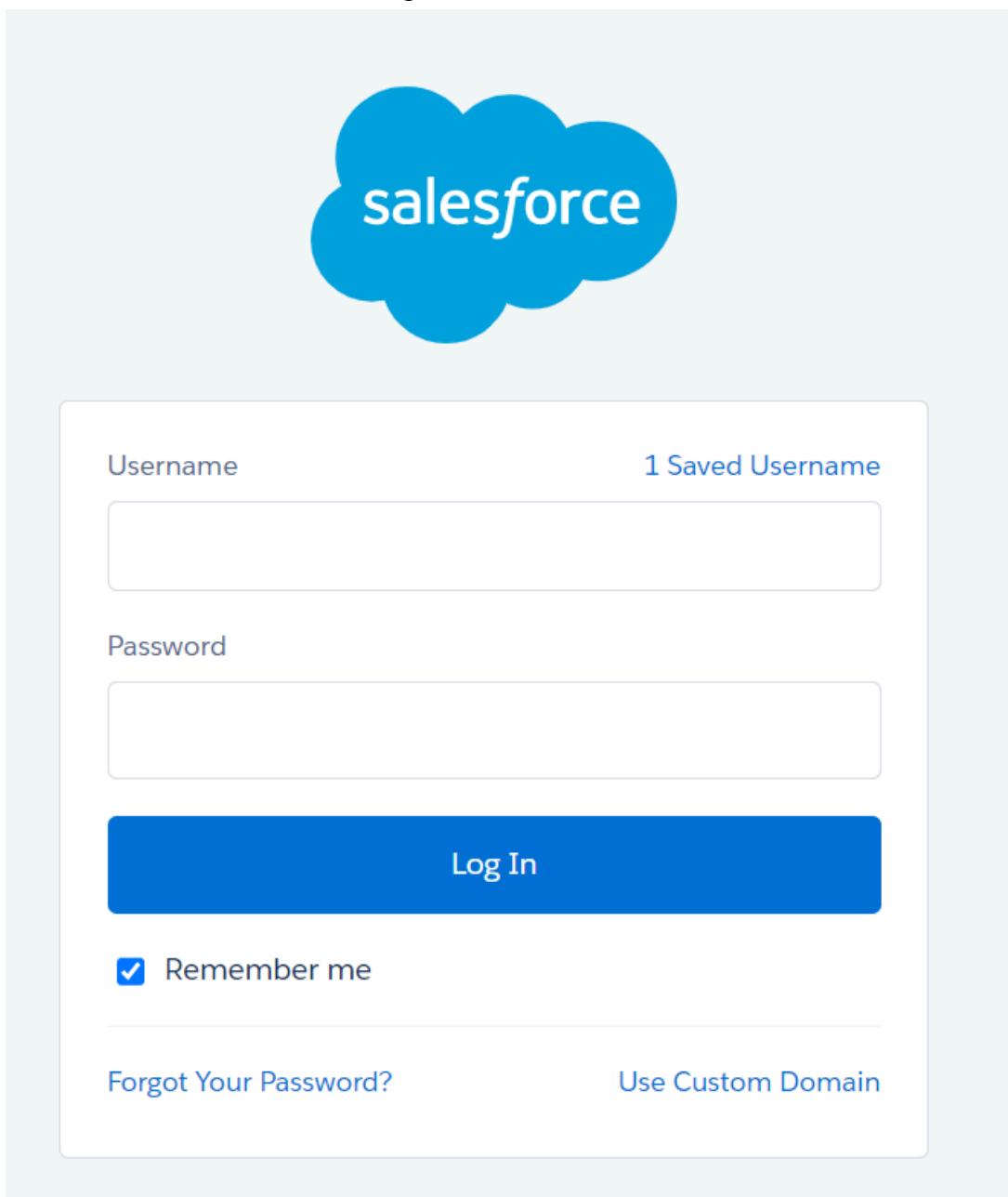
4. For this demonstration, I used the developer edition, hence it is Production.

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

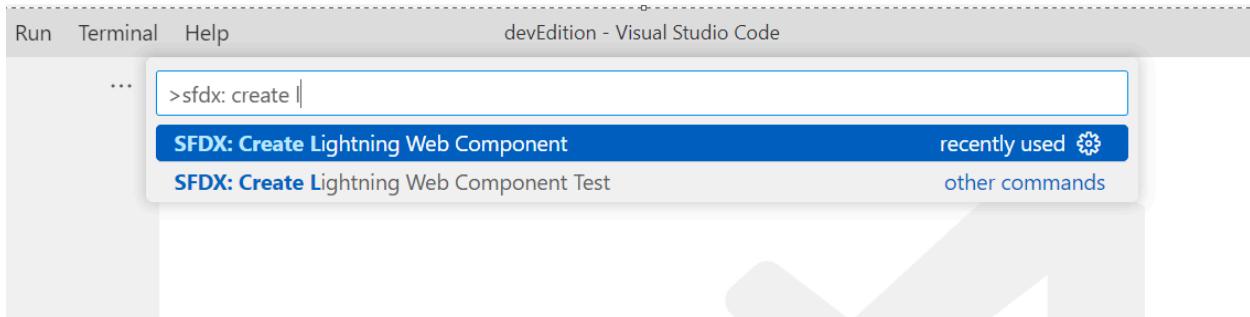


8. It will be successfully authorized.

#### **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.

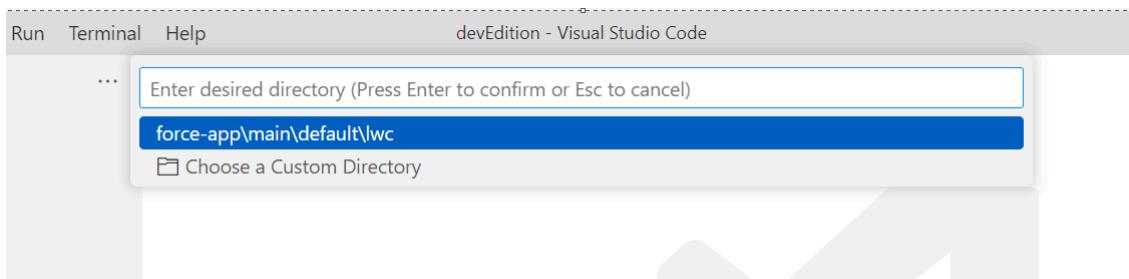
A screenshot of the Visual Studio Code interface. The top menu bar shows 'File', 'View', 'Go', 'Run', and '...'. The title bar says 'devEdition - Visual Studio Code'. The command palette search bar contains 'InvoiceOpportunity'. The code editor below shows the generated LWC metadata XML:

```

<?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>

```

3. Choose the directory.



4. LWC is created successfully.

A screenshot of the Visual Studio Code interface. The top menu bar shows 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', and '...'. The title bar says 'LWC Project Practise'. The left sidebar shows the 'EXPLORER' view with a tree structure of files and folders. The 'invoiceOpportunity' folder is expanded, showing 'invoiceOpportunity.js', 'invoiceOpportunity.html', and 'invoiceOpportunity.js-meta.xml'. The code editor shows the contents of 'invoiceOpportunity.js-meta.xml':

```

<?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>

```

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.

```
File Edit Selection View Go Run ... 🔍 LWC Project Practise
EXPLORER invoiceOpportunity.js invoiceOpportunity.html invoiceOpportunity.js-meta.xml
LWC PROJECT PRACTISE
> customValidationiniwc
> DataBaseWizardButton
> editRecordComponent
> flowInputs
> forms
> getAllThePicklistValuesOfAcc...
> getRecordsWireAdapter
> helloWorld
> insertContactAccount
> invoiceOpportunity
  invoiceOpportunity.html
  invoiceOpportunity.js
  invoiceOpportunity.js-meta.xml
> javascript
> lightningViewRecordCompon...
> looping
> navigateToHomePage
> navigateToNewRecordPage
> navigateToRecordRelati...
force-app > main > default > lwc > invoiceOpportunity > invoiceOpportunity.js > InvoiceOpportunity
1 import { LightningElement, api, track, wire } from 'lwc';
2 import getInvoices from '@salesforce/apex/OpportunityInvoiceswithLWC.getInvoices';
3 export default class InvoiceOpportunity extends LightningElement {
4   @api recordId;
5   @track invoiceCollection
6   cols = [
7     {label:"ID" , fieldName:'Name'},
8     {label:"Opportunity Id" , fieldName:'Opportunity__c'},
9     {label:"Quantity" , fieldName:'Quantity__c'},
10    {label:"Unit Price" , fieldName:'Unit_Price__c'},
11    {label:"Total Price" , fieldName:'Total_Price__c'},
12    {label:"Purchase Date" , fieldName:'Purchase_Date__c'}
13  ]
14  @wire(getInvoices,{OpportunityId:$recordId})
15  invoicefunction({data,error}){
16    console.log(this.recordId +'this is record Id');
17    if(data){
18      console.log(data);
19      this.invoiceCollection = data
20    }if(error){
21      console.log('this is error')
22      console.log(error);
23    }
24  }
25
26 }
```

### 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
```

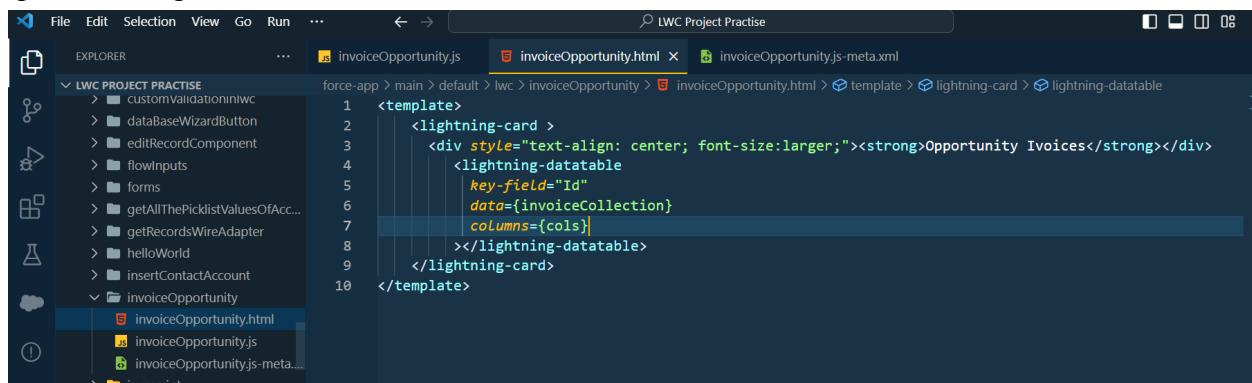
```

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.



The screenshot shows the LWC Project Practise code editor interface. The left sidebar displays a file tree under 'LWC PROJECT PRACTISE' with various components like customValidationInIWC, DataBaseWizardButton, editRecordComponent, flowInputs, forms, getAllThePicklistValuesOfAcc..., getRecordsWireAdapter, helloWorld, insertContactAccount, and invoiceOpportunity. The right pane shows the code for 'invoiceOpportunity.html'. The code includes an `<template>` block containing an `<lightning-card>` component. Inside the card, there's a `<div style="text-align: center; font-size:larger;"><strong>Opportunity Ivoices</strong></div>`. Below this, there's a `<lightning-datatable>` component with attributes `key-field="Id"`, `data={invoiceCollection}`, and `columns={cols}`.

```

<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>

```

### HTML File Code:

```

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

```

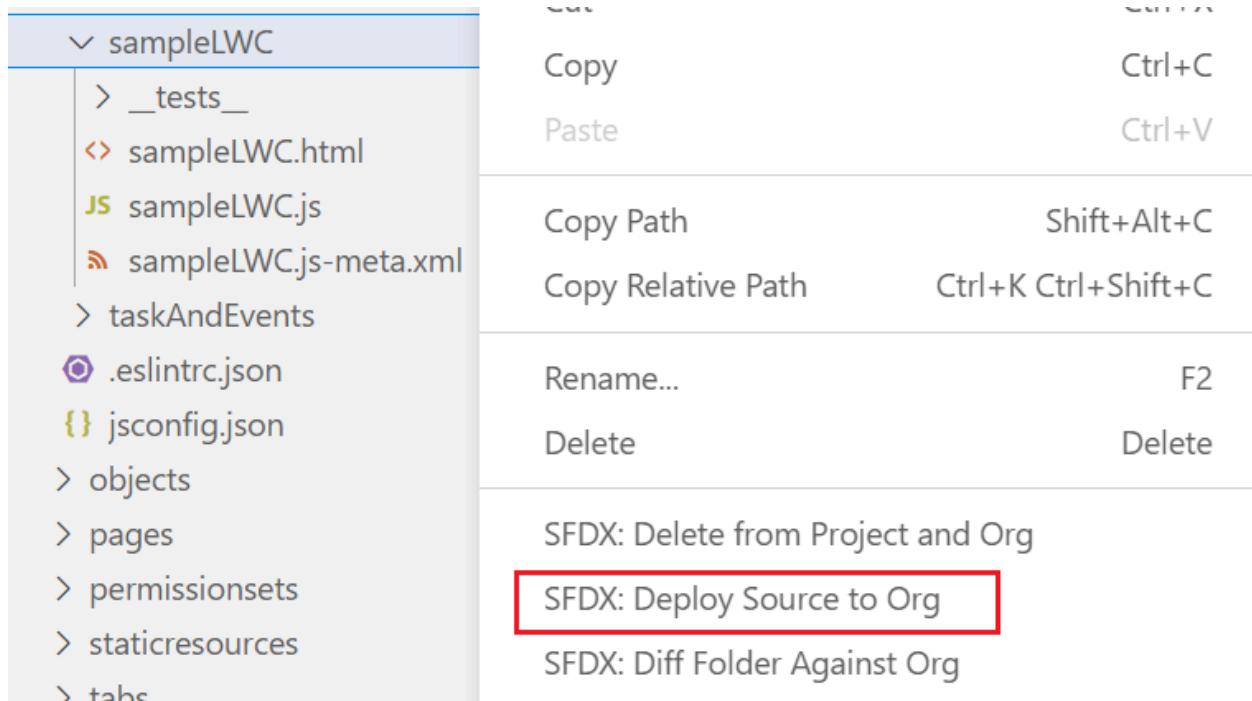
```

<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

The screenshot shows the VS Code Output tab with the following content:

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Salesforce CLI
==== Deployed Source
STATE FULL NAME TYPE PROJECT PATH
changed invoiceOpportunity LightningComponentBundle force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.html
changed invoiceOpportunity LightningComponentBundle force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js
changed invoiceOpportunity LightningComponentBundle force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js-meta.xml
10:05:04.331 ended SFDX: Deploy Source to Org

```

A message at the bottom of the output tab, '10:05:04.331 ended SFDX: Deploy Source to Org', is highlighted with a red box.

### 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.

The screenshot shows the Salesforce Object Manager interface for the Opportunity object. The left sidebar has a 'Buttons, Links, and Actions' section highlighted with a red box. The main content area displays a table of existing actions, and a red box highlights the 'New Action' button in the top right corner of this area.

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

The screenshot shows the 'New Action' configuration page for the Opportunity object. The 'Action Type' dropdown is set to 'Lightning Web Component' and the component is 'c.invoiceOpportunity'. The 'Name' field is filled with 'Invoices'. A red box highlights the 'Save' button at the bottom right.

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.

The screenshot shows the Salesforce setup interface for managing object layouts. The 'Opportunity' object is selected. In the left sidebar, 'Page Layouts' is highlighted. The main area displays a table of page layouts, with one entry, 'Opportunity Layout', highlighted by a red box.

PAGE LAYOUT NAME	CREATED BY	MODIFIED BY
Opportunity (Marketing) Layout	Mohammad Sameer, 22/11/2023, 2:19 pm	Mohammad Sameer, 28/11/2023, 9:45 am
Opportunity (Sales) Layout	Mohammad Sameer, 22/11/2023, 2:19 pm	Mohammad Sameer, 28/11/2023, 9:45 am
Opportunity (Support) Layout	Mohammad Sameer, 22/11/2023, 2:19 pm	Mohammad Sameer, 28/11/2023, 9:45 am
<b>Opportunity Layout</b>		Mohammad Sameer, 01/12/2023, 10:57 am

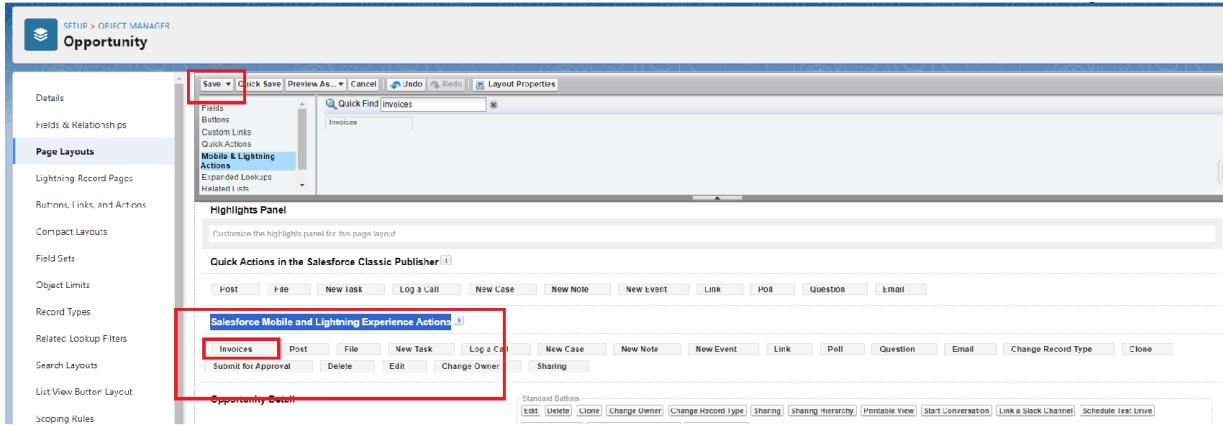
3. Click on Mobile And Lightning Action as show on below Image

The screenshot shows the 'Opportunity Layout' configuration screen. The 'Mobile & Lightning Actions' section is highlighted by a red box. This section contains various buttons for actions like Post, File, New Task, Log a Call, etc., categorized under 'Salesforce Mobile and Lightning Experience Actions'.

4. Search for invoice on Quick Find.

The screenshot shows the 'Opportunity Layout' configuration screen again. The 'Quick Find' search bar at the top is populated with the word 'Invoices', and the results list shows 'Invoices'. Below the search bar, the 'Salesforce Mobile and Lightning Experience Actions' section is expanded, also highlighted with a red box. This expanded view shows more detailed options for mobile and lightning actions.

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



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 :

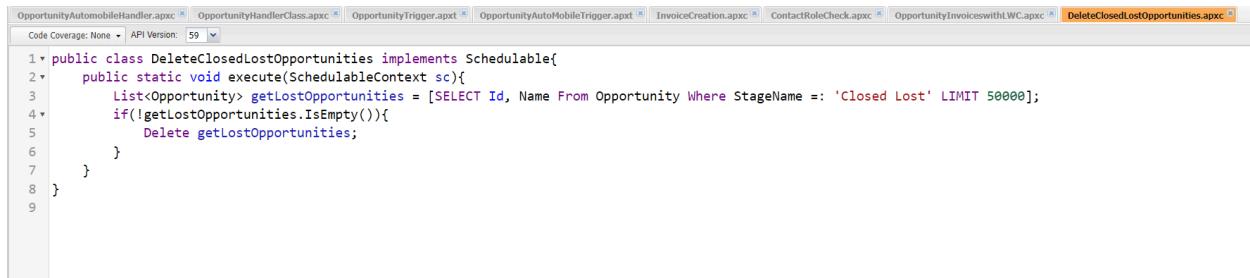
```
public class SomeClass implements Schedulable {  
    public void execute(SchedulableContext ctx) {  
        // awesome code here  
    }  
}
```

#### 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 :



```

OpportunityAutomobileHandler.apxc OpportunityHandlerClass.apxc OpportunityTrigger.apxt OpportunityAutoMobileTrigger.apxt InvoiceCreation.apxc ContactRoleCheck.apxc OpportunityInvoiceswithLWC.apxc DeleteClosedLostOpportunities.apxc
Code Coverage: None API Version: 59
1 public class DeleteClosedLostOpportunities implements Schedulable{
2     public static void execute(SchedulableContext sc){
3         List<Opportunity> getLostOpportunities = [SELECT Id, Name From Opportunity Where StageName =: 'Closed Lost' LIMIT 50000];
4         if(!getLostOpportunities.IsEmpty()){
5             Delete getLostOpportunities;
6         }
7     }
8 }

```

```

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.

The screenshot shows the Salesforce setup interface. A search bar at the top contains the text "apex classes". Below the search bar, under the "Custom Code" section, there is a yellow-highlighted link labeled "Apex Classes". A message below the search bar says, "Didn't find what you're looking for? Try using Global Search." At the bottom of the page, there is a navigation bar with links like "Setup", "Home", "Object Manager", and a "Help for this Page" button.

**Apex Classes**

Apex Code is an object oriented programming language that allows developers to develop on-demand business applications on the Lightning Platform.

Action	Name	Namespace Prefix	API Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit   Del   Security	ContractRoleCheck		59.0	Active	672	Mohammad Sameer	29/11/2023, 10:13 am
Edit   Del   Security	DeleteClosedLostOpportunities		59.0	Active	356	Mohammad Sameer	01/12/2023, 11:57 am
Edit   Del   Security	InvoiceCreation		59.0	Active	1,243	Mohammad Sameer	29/11/2023, 10:05 am
Edit   Del   Security	OpportunityAutomobileHandler		59.0	Active	1,085	Mohammad Sameer	29/11/2023, 10:12 am
Edit   Del   Security	OpportunityHandlerClass		59.0	Active	4,041	Mohammad Sameer	29/11/2023, 9:49 am
Edit   Del   Security	OpportunityInvoicesWithLWC		59.0	Active	319	Mohammad Sameer	29/11/2023, 3:04 pm

**Dynamic Apex Classes**

- Click on Schedule Apex and enter the Job name.

- Job Name : DeleteOpportunitySchedule

**Schedule Apex**

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

		Save	Cancel
Job Name	DeleteOpportunitySchedule		
Apex Class	DeleteClosedLostOpportuni 		
Schedule Apex Execution			
<div style="display: flex; align-items: center;"> <span>Frequency</span> <input checked="" type="radio"/> Weekly           <input type="radio"/> Monthly           <div style="margin-left: 20px;">             Recurs every week on               <input type="checkbox"/> Sunday             <input checked="" type="checkbox"/> Monday             <input type="checkbox"/> Tuesday             <input type="checkbox"/> Wednesday             <input type="checkbox"/> Thursday             <input type="checkbox"/> Friday             <input type="checkbox"/> Saturday           </div> </div>			
Start	01/12/2023	[ 01/12/2023 ]	
End	01/01/2024	[ 01/12/2023 ]	
Preferred Start Time	10:00 am 		
Exact start time will depend on job queue activity.			

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

Apex Classes ~ Salesforce - Developer Edition - Google Chrome

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

**Lookup**



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

< [Clear Search Results](#)

**Search Results**

Name	Namespace Prefix	Api Version
DeleteClosedLostOpportunities		59

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

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

**Schedule Apex**

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

Job Name: DeleteOpportunitySchedule  
Apex Class: DeleteClosedLostOpportunit

Schedule Apex Execution

Frequency:  Weekly  Monthly

Recur every week on:

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

Start: 01/12/2023 [ 01/12/2023 ]  
End: 01/01/2024 [ 01/12/2023 ]  
Preferred Start Time: 10:00 am

Exact start time will depend on job queue activity.

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

SETUP Scheduled Jobs

All Scheduled Jobs

The All Scheduled Jobs page lists all of the jobs scheduled by your users. Multiple job types may display on this page. You can delete scheduled jobs if you have the permission to do so.

View: All Scheduled Jobs | Create New View

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type
Manage   Del	DeleteOpportunitySchedule	Sameer_Mohammad	01/12/2023, 12:02 pm		04/12/2023, 10:00 am	Scheduled Apex
Del	Metalitics Data Loader Job for Org: 00DSj00000DwHvY	User_Interaction	22/11/2023, 2:21 pm	30/11/2023, 10:32 pm	01/12/2023, 10:32 pm	Autonomous Data Loader Job

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

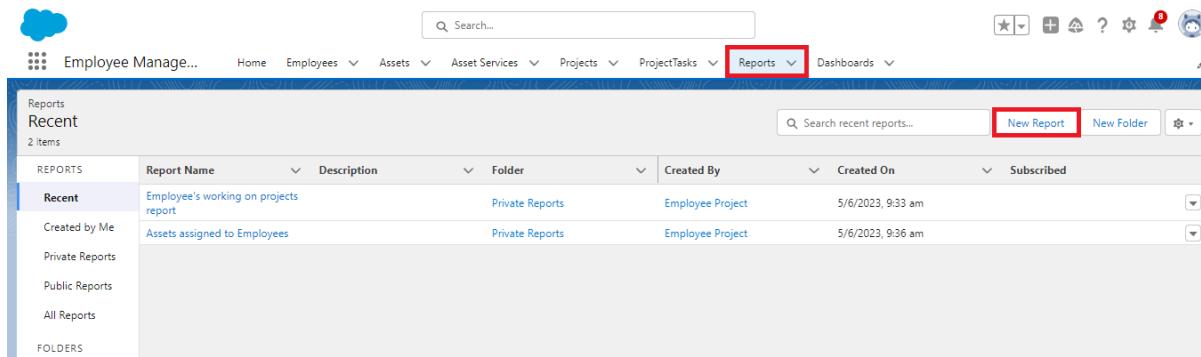
Use Case:

The CEO of an organization wants to have brief data on Opportunity Sales along with Invoices generated.

Let's create a Report.

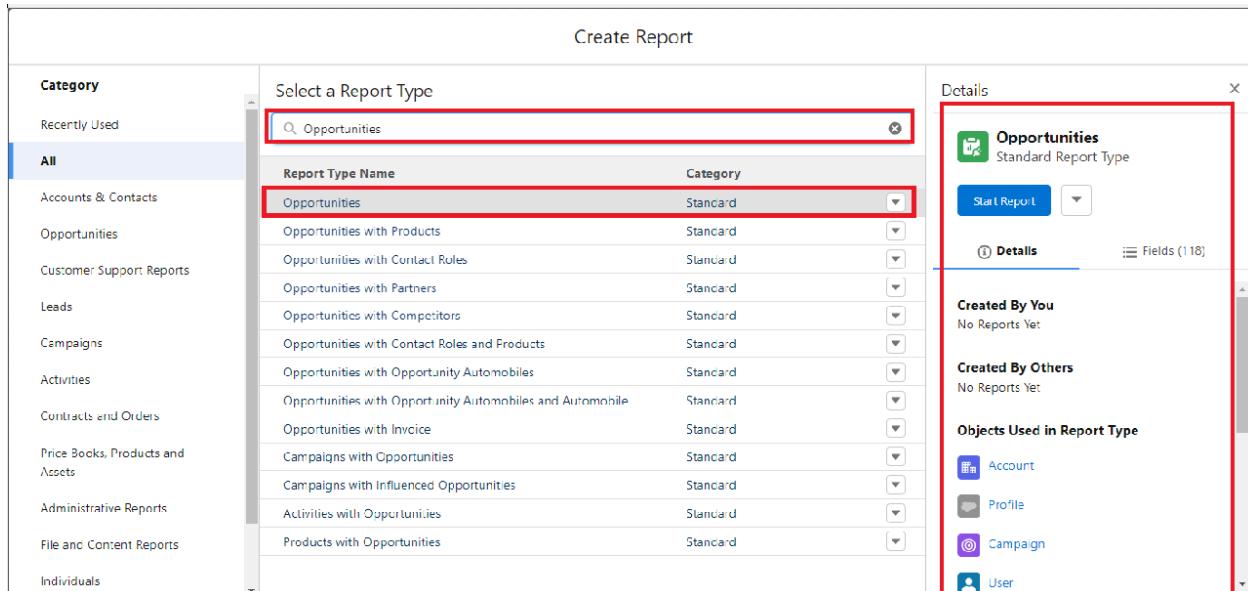
## Activity- 1 : Create Report on Opportunity

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



The screenshot shows the application's main navigation bar with various tabs like Home, Employees, Assets, Asset Services, Projects, ProjectTasks, Reports, and Dashboards. The 'Reports' tab is highlighted with a red box. Below the navigation is a search bar and a toolbar with icons for star, refresh, help, and other functions. The main content area is titled 'Reports' and 'Recent'. It lists two reports: 'Employee's working on projects report' and 'Assets assigned to Employees'. A sidebar on the left shows categories like Recent, Created by Me, Private Reports, Public Reports, and All Reports. At the bottom left is a 'FOLDERS' section.

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



The screenshot shows the 'Create Report' dialog. On the left is a sidebar with categories like Recently Used, Accounts & Contacts, Opportunities, Customer Support Reports, Leads, Campaigns, Activities, Contracts and Orders, Price Books, Products and Assets, Administrative Reports, File and Content Reports, and Individuals. The 'Opportunities' category is selected. In the center, a search bar contains 'Opportunities' with a red box around it. Below the search bar is a table with columns 'Report Type Name' and 'Category'. The first row, 'Opportunities', is selected and highlighted with a red box. To the right is a 'Details' panel for the 'Opportunities' report type. It includes sections for 'Opportunities' (Standard Report Type), 'Start Report' button, 'Details' (with 'Created By You' and 'Created By Others' sections), and 'Objects Used in Report Type' (Account, Profile, Campaign, User).

4. Customize your report

- Add fields from left pane as shown below

**New Employees Report**

**Opportunity Closed Won Report**

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”.

The screenshot shows the Zoho Reports interface for creating a report titled "Automobile Information". The "Fields" panel on the left is highlighted with a red box, displaying a list of fields: Name Of Manufacturer, Model, Built date, Total Number of Cylinders, Colour, Quantity, Price, and VIN. The main area shows a preview of the report data with 10 rows of automobile information, including details like manufacturer (Toyota, Ford, Subaru, Hyundai, Nissan, Audi, Mercedes-Benz, BMW, Chevrolet), model (Corolla, Mustang, Outback, Sonata, Altima, A4, C-Class, 3 Series, Malibu), built date (15-05-2022 to 30-11-2022), total cylinders (4 to 6), colour (Red, Blue, Green, Red, Silver, Blue, Gray, White, Black), quantity (12 to 33), price (₹20.00 to ₹26.00), and VIN numbers.

Automobile Information: Name Of Manufacturer	Model	Built date	Total Number of Cylinders	Colour	Quantity	Price	VIN
Toyota	Corolla	15-05-2022	4	Red	12	₹20.00	1IGCM0B633A004352
Ford	Mustang	10-01-2023	8	Blue	54	₹35.00	2C3CDZAG4KH123456
Subaru	Outback	14-10-2023	6	Green	56	₹30.00	5JTFEH51EL123456
Hyundai	Sonata	08-06-2022	4	Red	78	₹26.00	TG1YY32G55123456
Nissan	Altima	25-07-2023	4	Silver	77	₹24.00	2T2H431U45C123456
Audi	A4	12-08-2022	4	Blue	9	₹33.00	JN1EK13R7XJ123456
Mercedes-Benz	C-Class	18-09-2023	4	Gray	24	₹38.00	JN1B11CPXHW123456
BMW	3 Series	05-04-2023	6	White	116	₹42.00	WA1VAAF74KD123456
Chevrolet	Malibu	30-11-2022	6	Black	33	₹28.00	SV3E1EA3KF123456
					459	₹276.00	

Filters :-

The screenshot shows the Zoho Reports interface with filters applied to the "Automobile Information" report. The "Filters" panel on the left is highlighted with a blue box, showing a dropdown menu with options like "All automobile information" and "Automobile Information: Created Date All Time". The main area displays the same report data as the previous screenshot, showing 10 rows of automobile information with columns for Name Of Manufacturer, Model, Built date, Total Number of Cylinders, Colour, Quantity, Price, and VIN.

Automobile Information: Name Of Manufacturer	Model	Built date	Total Number of Cylinders	Colour	Quantity	Price	VIN
Toyota	Corolla	15-05-2022	4	Red	12	₹20.00	1IGCM0B633A004352
Ford	Mustang	10-01-2023	8	Blue	54	₹35.00	2C3CDZAG4KH123456
Subaru	Outback	14-10-2023	6	Green	56	₹30.00	5JTFEH51EL123456
Hyundai	Sonata	08-06-2022	4	Red	78	₹26.00	TG1YY32G55123456
Nissan	Altima	25-07-2023	4	Silver	77	₹24.00	2T2H431U45C123456
Audi	A4	12-08-2022	4	Blue	9	₹33.00	JN1EK13R7XJ123456
Mercedes-Benz	C-Class	18-09-2023	4	Gray	24	₹38.00	JN1B11CPXHW123456
BMW	3 Series	05-04-2023	6	White	116	₹42.00	WA1VAAF74KD123456
Chevrolet	Malibu	30-11-2022	6	Black	33	₹28.00	SV3E1EA3KF123456
					459	₹276.00	

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.

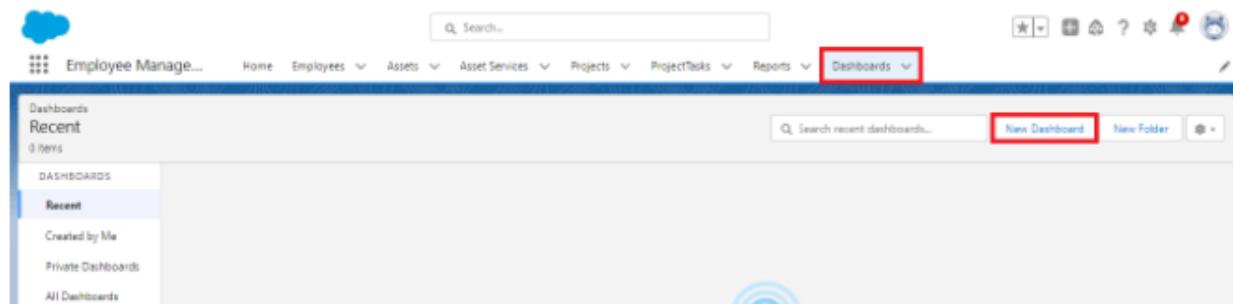
Use Case:

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.

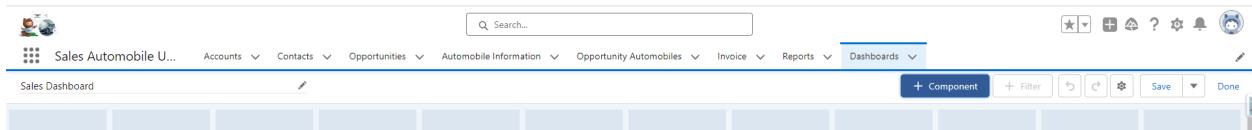


2. Give a Name and click on Create.

The screenshot shows the 'New Dashboard' creation form. It has fields for 'Name' (containing 'Dashboard 1'), 'Description' (empty), 'Folder' (set to 'Private Dashboards'), and a 'Select Folder' button. At the bottom are 'Cancel' and 'Create' buttons, with 'Create' highlighted with a red box.

Name : Automobile Sales

3. Select add component.



4. Select a Report and click on select.

Select Report

Reports	Select Report
<b>Recent</b>	<input type="text" value="Q. Search Reports and Folders..."/> <span style="float: right;">Reports and Folders ▾</span>
Created by Me	Opportunity With Automobile Data Mohammad Sameer - 01-Dec-2023, 12:52 pm - Public Reports
Private Reports	
Public Reports	Opportunity Closed Won Report Mohammad Sameer - 01-Dec-2023, 12:21 pm - Public Reports
All Reports	Automobile Information Report Mohammad Sameer - 01-Dec-2023, 12:37 pm - Public Reports
<b>Folders</b>	
Created by Me	Sample Flow Report: Screen Flows Automated Process - 22-Nov-2023, 2:19 pm - Public Reports
Shared with Me	
All Folders	

Cancel Select

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

The Created Dashboard will look like this.

The dashboard displays three reports:

- Opportunity With Automobile Data:** A bar chart showing the sum of quantity for different account names. The data is as follows:
 

Account Name	Sum of Quantity
Burlin...	2
Edge ...	34
Unite...	126
Test	8
- Opportunity Closed Won Report:** A bar chart showing the record count for different account names. The data is as follows:
 

Account Name	Record Count
Burlington T...	1
Dickenson plc	1
Edge Comm...	1
Extreme Log...	4
GenePoint	3
Grand Hotel...	3
Pyramid Co...	5
United Oil &...	10
University of...	3
- Automobile Information Report:** A table showing automobile information. The data is as follows:
 

Automobile Information: Name Of Manufactur...	Model	Built Date	Total Number of C...	Color
Audi	A4	12-08	4	Blue
BMW	3 Serie	05-04	6	White
Chevrolet	Mallibu	30-11	6	Black
Ford	Mustang	10-01	8	Blue

\*\*\*\*\*END\*\*\*\*\*