**Inventory Management System**

ProblemStatementMany organizations struggle to efficiently manage their inventory, track stock levels across multiple warehouses, and process orders in a timely manner. Manual inventory management often leads to errors such as stock discrepancies, delayed order fulfilment, overstocking, or stockouts. Additionally, communication gaps between suppliers, inventory managers, and sales teams can result in operational inefficiencies and missed business opportunities.  
  
The Inventory and Order Management System aim to address these challenges by providing a centralized Salesforce-based solution that automates stock tracking, manages orders, sends low-stock alerts, and streamlines approval workflows. This ensures accurate inventory visibility, faster order processing, and improved decision-making for businesses.

Phase – 1 Problem Understanding & Industry Analysis

**Requirement Gathering:**

* Track products across multiple warehouses with real-time visibility.
* Automate order fulfilment and restock requests.
* Provide low-stock alerts and approval workflows for large purchases.
* Ensure scalability, user-friendliness, and automation in the system.
* Support integrations such as barcode scanning, payment gateways, and Google Maps for order routing.

**Stakeholder Analysis**

* Warehouse Managers → need real-time stock visibility and restock alerts.
* Procurement Team → requires supplier tracking and purchase order automation.
* Sales Team → depends on live stock data to avoid overselling.
* Customers (B2B/B2C) → expect self-service portals to place orders and track deliveries.
* Management/Executives → demand dashboards for supplier performance, trends, and predictive analytics.

**Business Process Mapping**

* Current process: Procurement → Storage → Order Placement → Order Routing → Shipment → Restock Request.
* Pain points: manual approvals, frequent stockouts, overstocking, lack of visibility across warehouses.
* Salesforce improvements: automation via flows, approval processes, dashboards.

**Industry-specific Use Case Analysis**

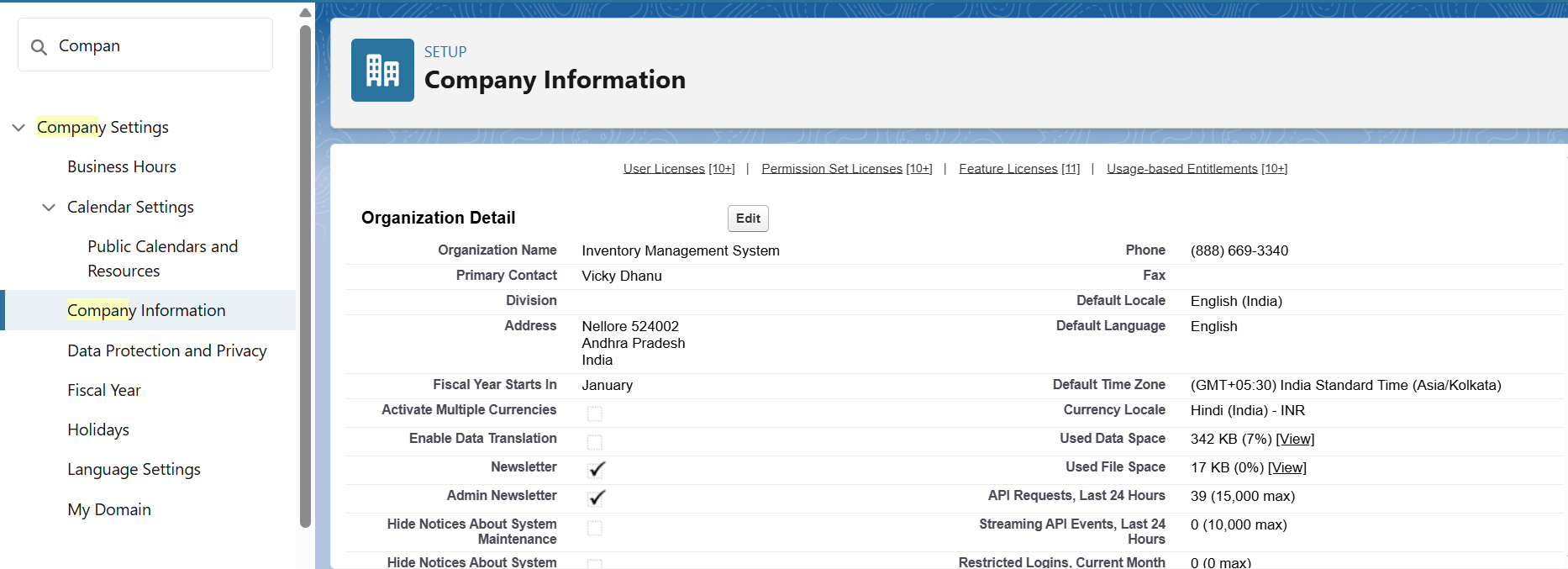
* Retail → stock management across multiple warehouses.
* Manufacturing → raw material tracking and supplier reliability monitoring.
* Healthcare/Pharma → expiry-date tracking for compliance and waste reduction.
* E-commerce → nearest fulfilment centre routing and real-time delivery tracking.

Phase 2: Org Setup & Configuration

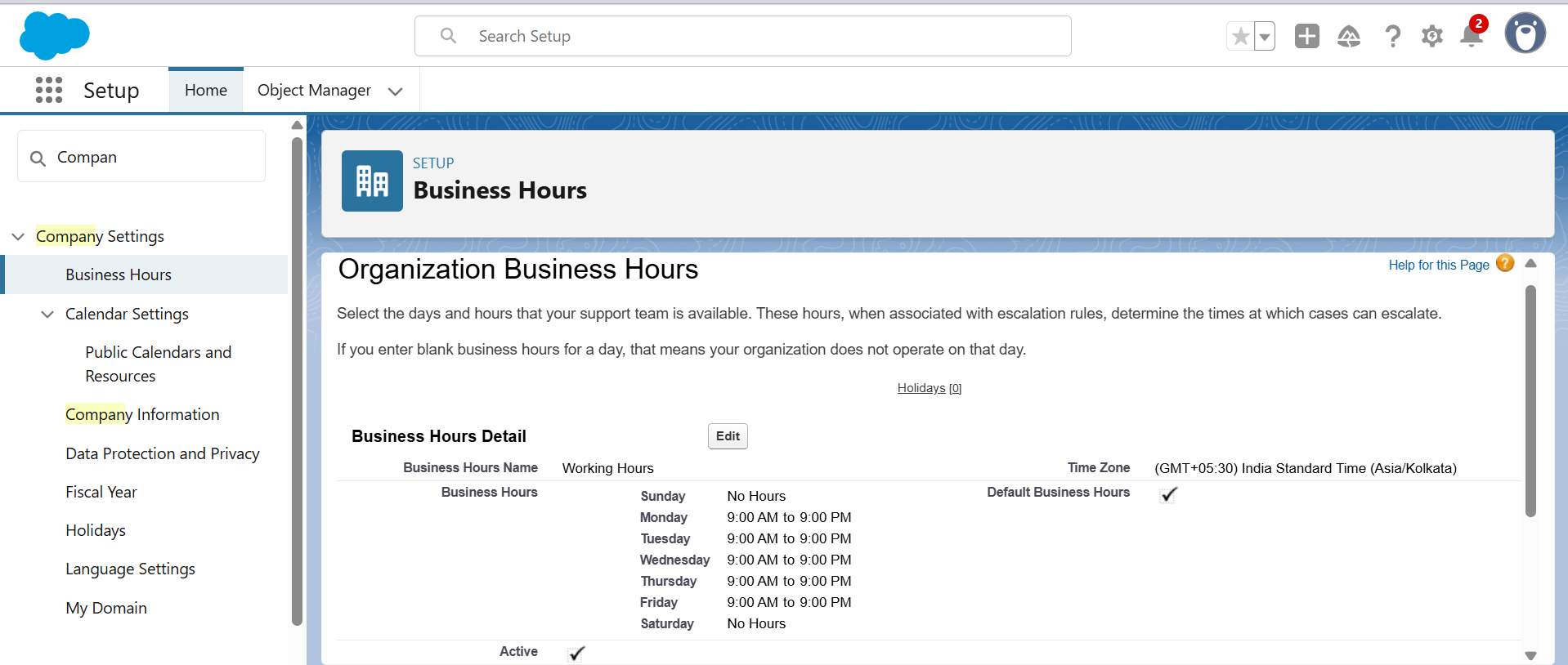
**Salesforce Editions:**

I have signed up for a new developer edition org for this project.

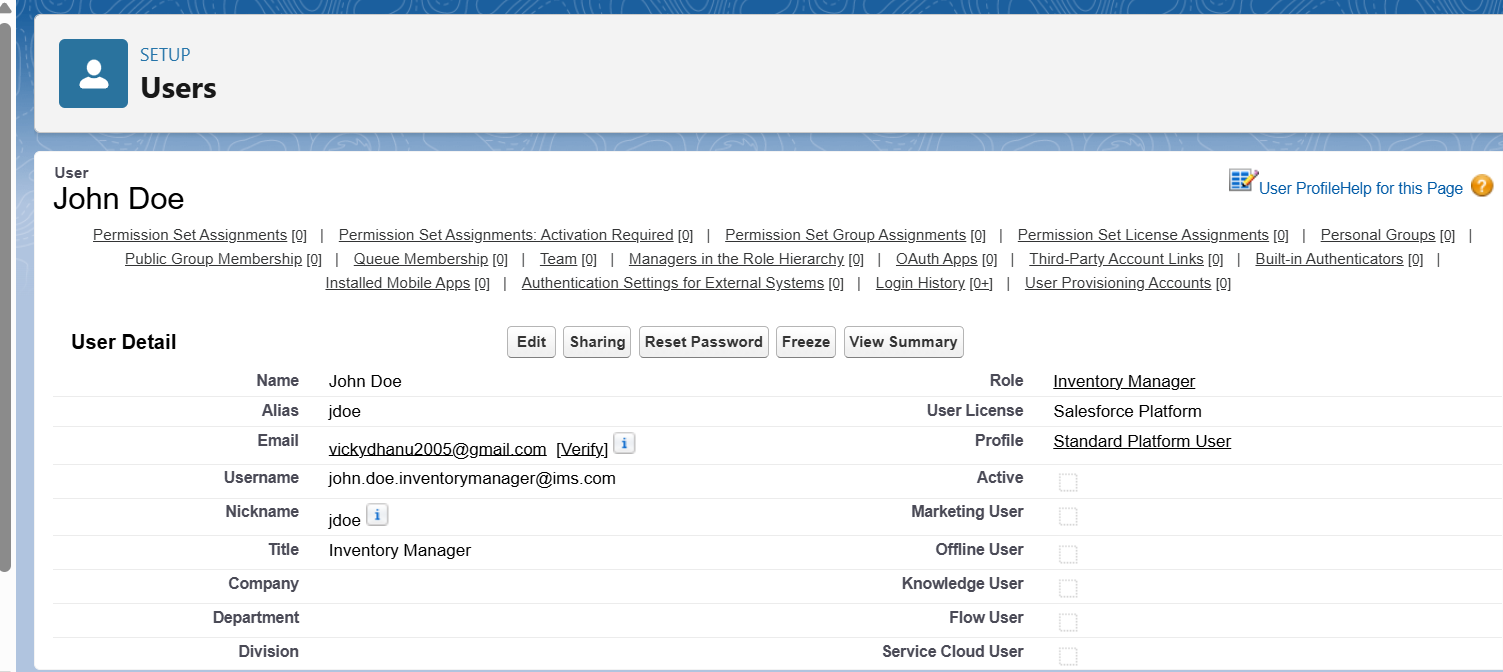
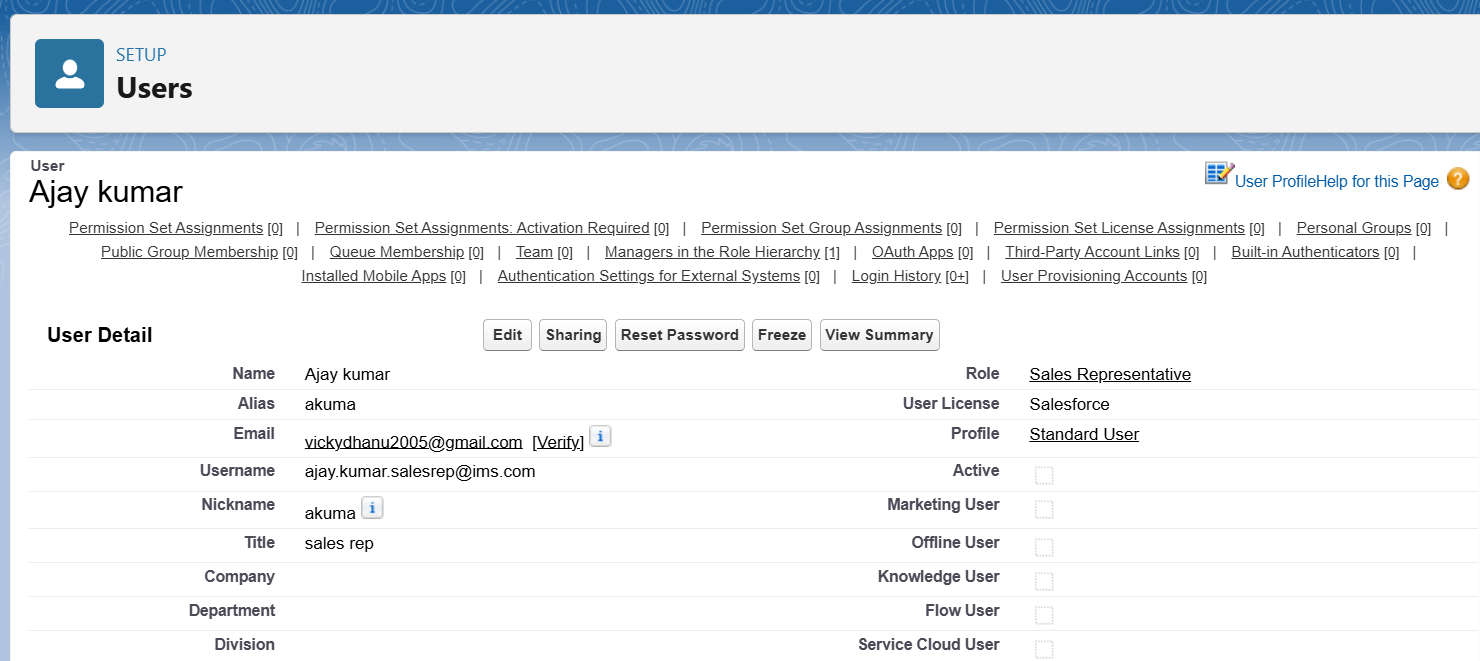
**Company Profile Setup:**

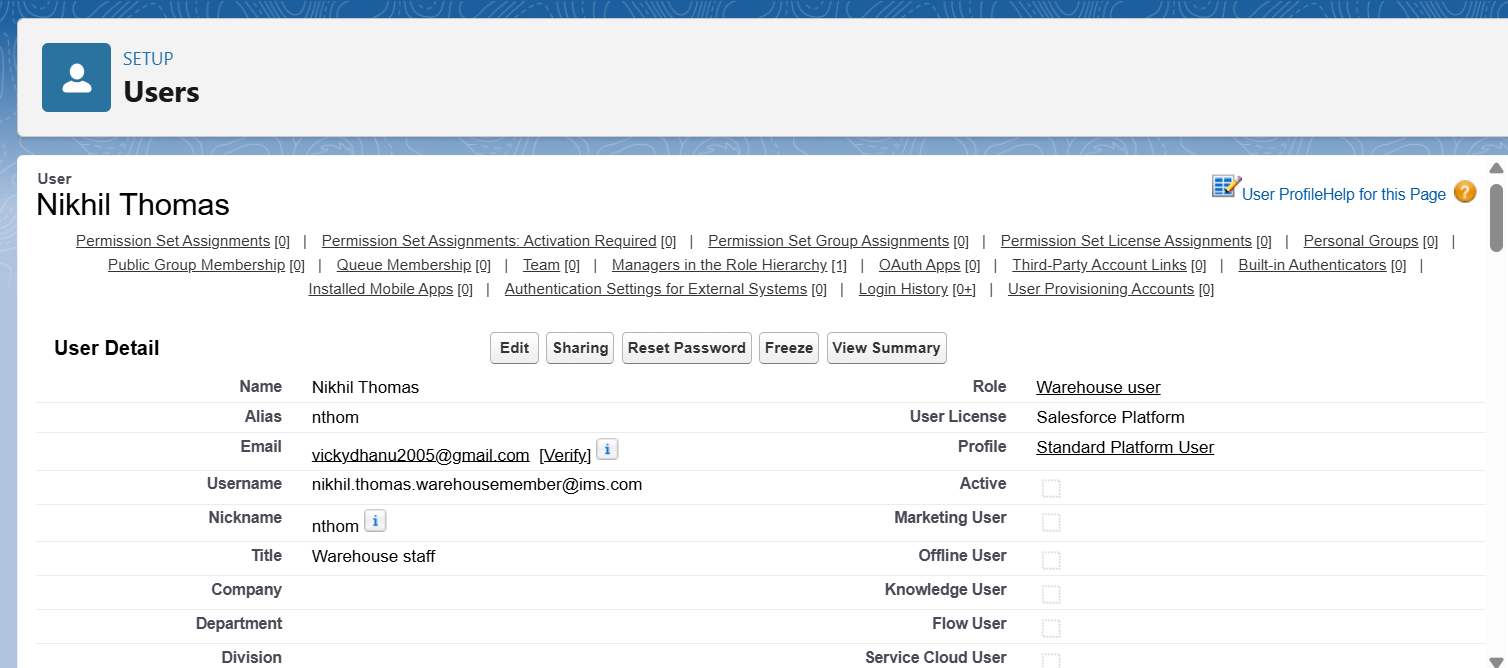
****

**Business Hours & Holidays:**

Our organization’s business hours  
 

**User Setup & Licenses:  
  
Steps for creating users:**

* Click the gear icon you will go into setup.
* In Quick Find search users.
* click on users.
* In the user’s page click on create new user button and fill all the details like first name, last name, Gmail, username, role and profile you wish to give.
* I have created 3 users John Doe, Ajay Kumar, Nikhil Thomas.
* For John Doe, I gave Inventory Manager role.
* He can create, edit, delete products, inventory stock, orders.
* Runs Inventory and stock level reports.
* Can view warehouse staff and sales representative activities.
* For Ajay Kumar, I gave sales rep role.
* He creates sales orders for customers.
* Checks product availability before confirming orders.
* Reports to the Inventory Manager.
* For Nikhil Thomas, I gave warehouse user role.
* Cannot create new products.
* Can update the stock count.
* I myself acting as administrator.
* I have access to all the records.
* I will take care about the data loading etc.
* Below are the details of every user
* John Doe  
  
* Ajay Kumar  
  
* Nikhil Thomas



**Profiles:**

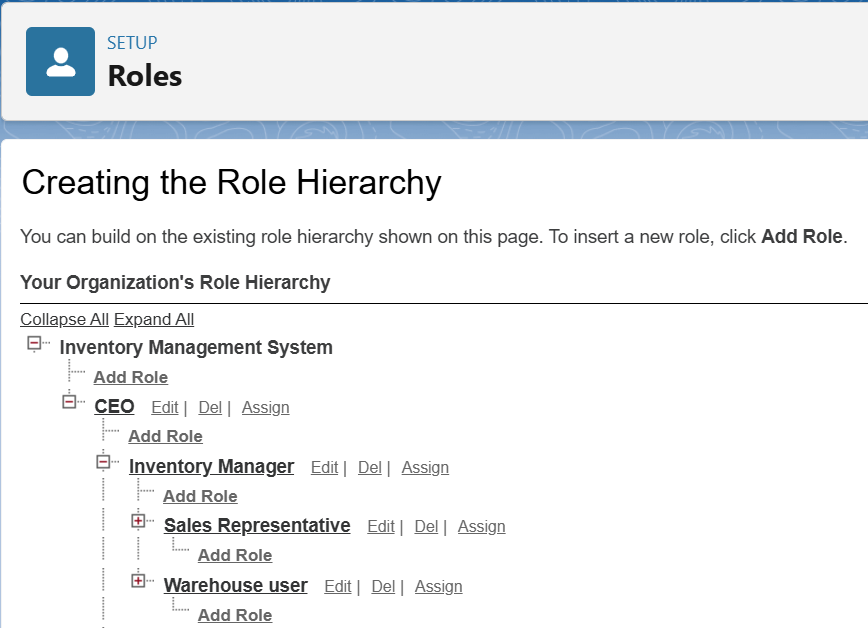
For the users I created I assigned the following profiles  
 Vicky Dhanu (Myself) – System Administrator  
 John Doe – Standard Platform User  
 Ajay Kumar – Standard User  
 Nikhil Thomas – Standard Platform User

**Roles:**

Following are the roles that I gave to my users of the project:

Vicky Dhanu – CEO / System Admin

John Doe – Inventory Manager  
 Ajay Kumar – Sales Rep  
 Nikhil Thomas – Warehouse staff member

Below is the screenshot representing the role hierarchy of the project  
   
 

Sales Rep and warehouse user reports to the Inventory Manager.  
Inventory Manager reports to CEO.

**OWD:**

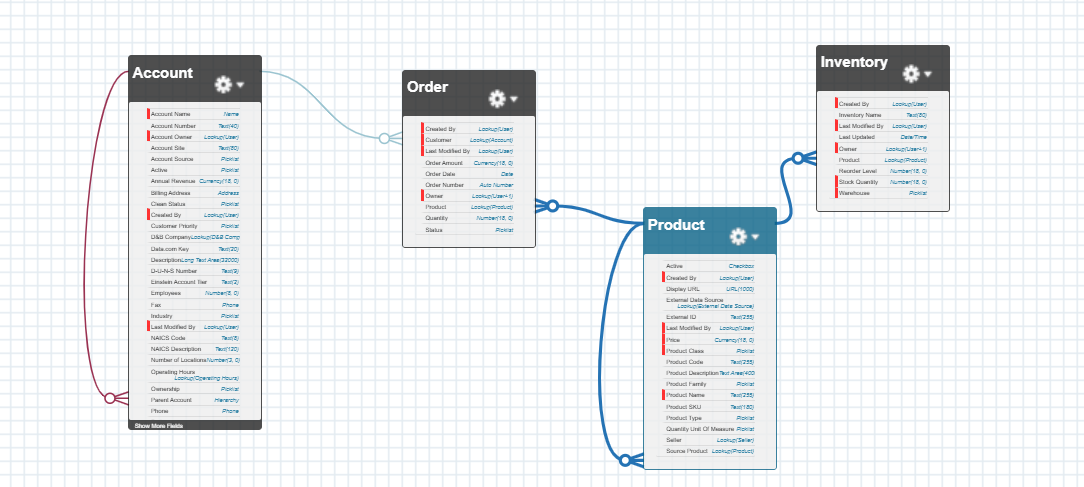
Below table gives details about the org wide defaults  
 Object OWD setting Explanation

|  |  |  |
| --- | --- | --- |
| Accounts | Private | Each Sales Rep should only see their own customers; managers see all via hierarchy. |
| Contacts | Private | Tied to Accounts; ensures customer information is secure per Sales Rep. |
| Orders (Custom) | Private | Orders created by Sales Reps are only visible to them; managers can see all. |
| Products | Public Read Only | All users should see available products, but only Inventory Manager/CEO can edit. |
| Inventory (Custom) | Public Read Only / Private | Sales Reps can view stock; Warehouse Staff & Inventory Manager can update. |
| Reports & Dashboards | Controlled by Parent | Visibility is determined by the access level of the underlying object data. |

Phase 3: Data Modelling & Relationships

**Standard & Custom Objects:**

* The standard objects that I used are Account and Product.
* The custom objects that I created are Inventory and Order.
* The below is the image of schema that I created using the schema builder



**Fields:**

* The fields that I used in the account object are:

Account Name  
 Account Number

Industry  
 Phone  
 Active  
 Shipping Address

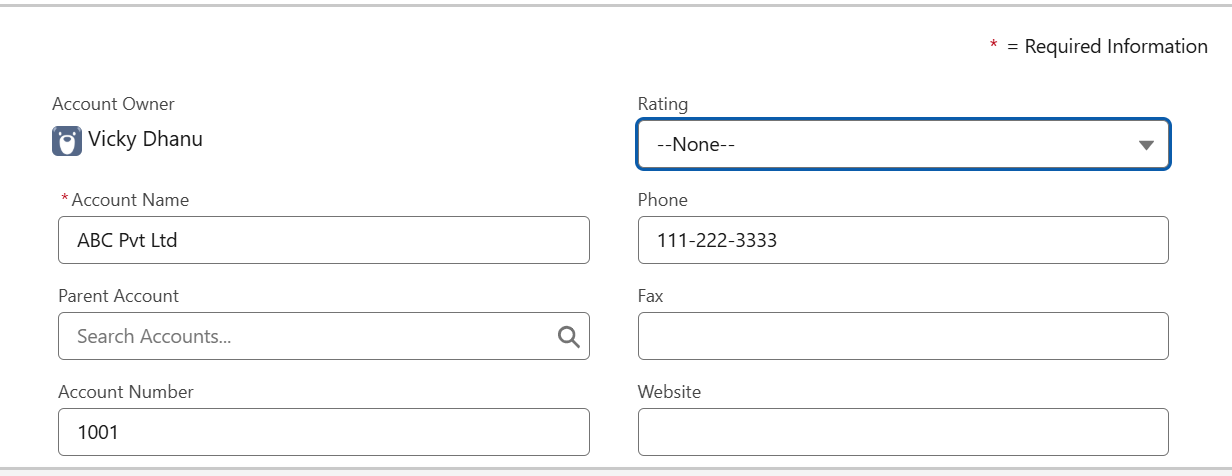
* The fields that I used in the product object are:  
   Product Name  
   Product Code  
   Price  
   Active  
   Product Family  
   Product Description
* The fields that I created in the inventory object are:

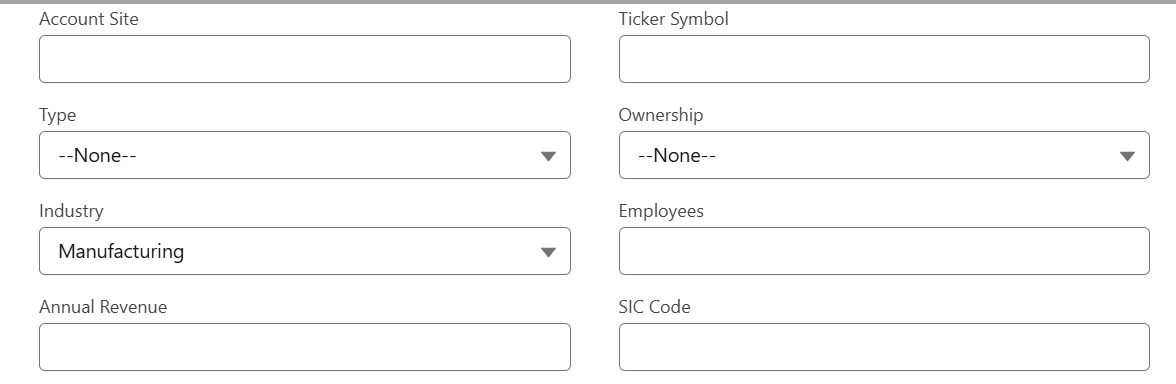
Inventory Name  
 Product  
 Stock Quantity  
 Reorder Level  
 Last Updated  
 Warehouse

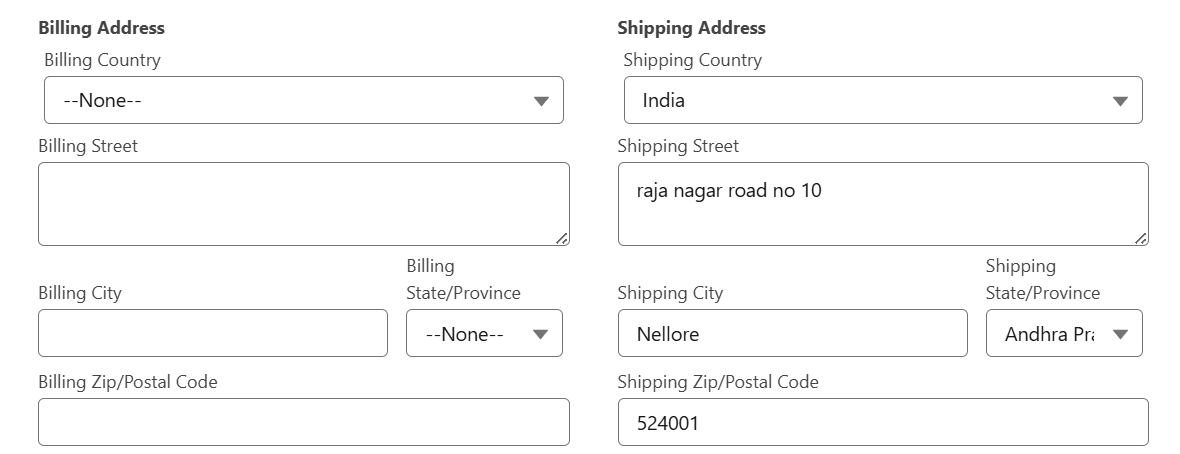
* The fields that I created in the order object are:

Order Number  
Customer  
Product  
Quantity  
Order Date  
Status  
Order Amount

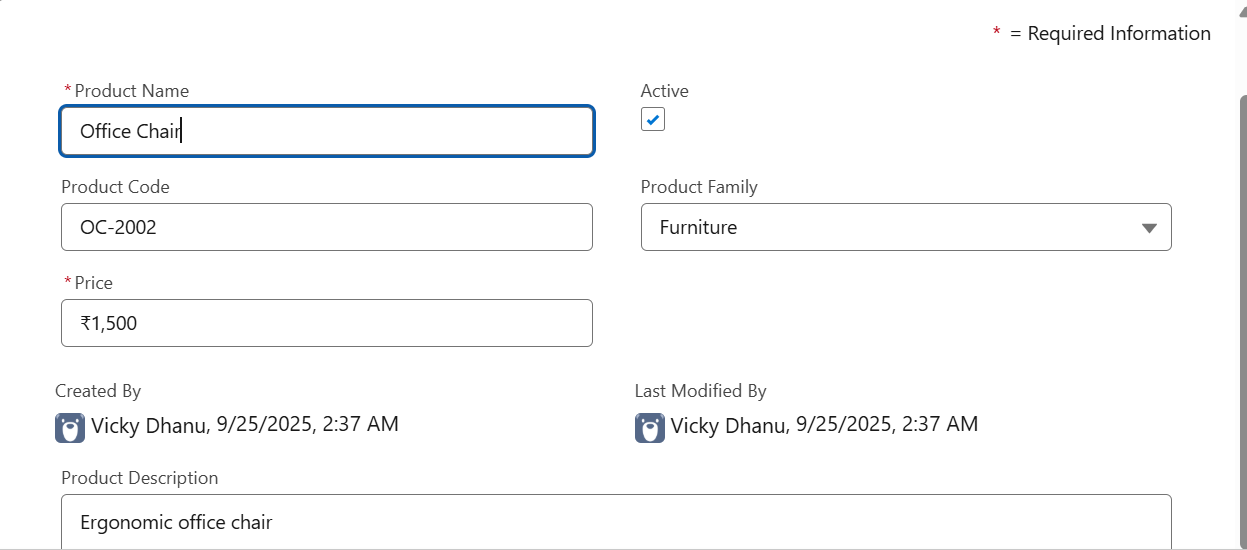
Images of record I created in the project for every object

* For Account Object:  
   

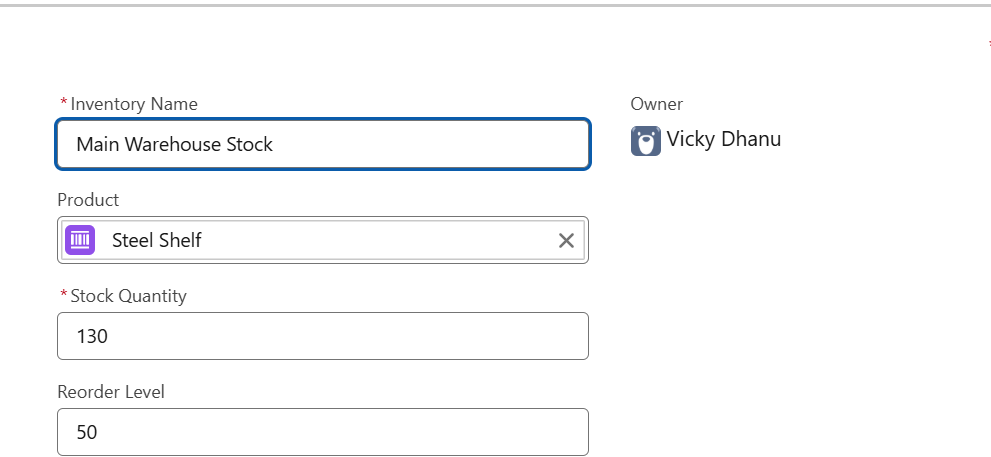


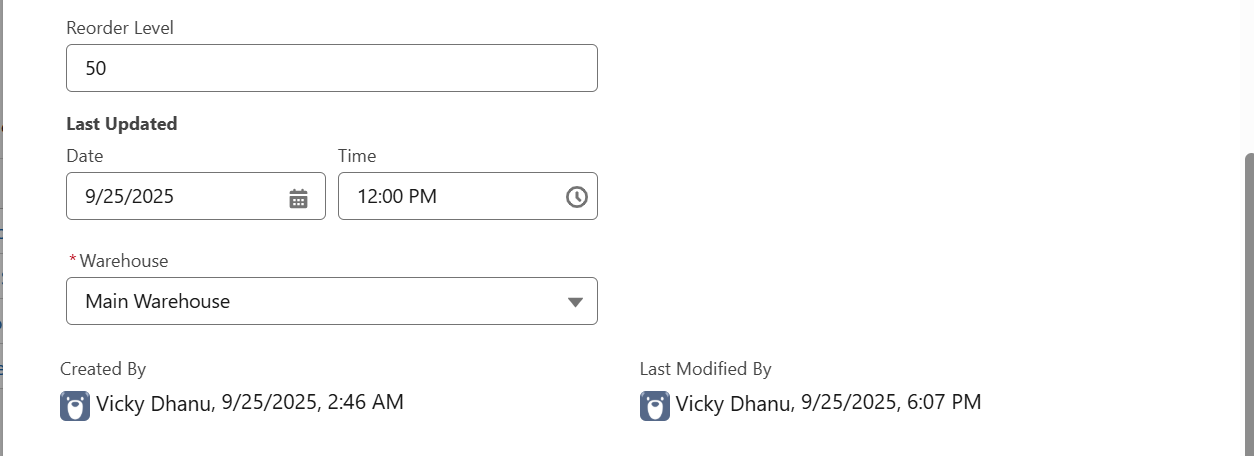


* For Product Object:



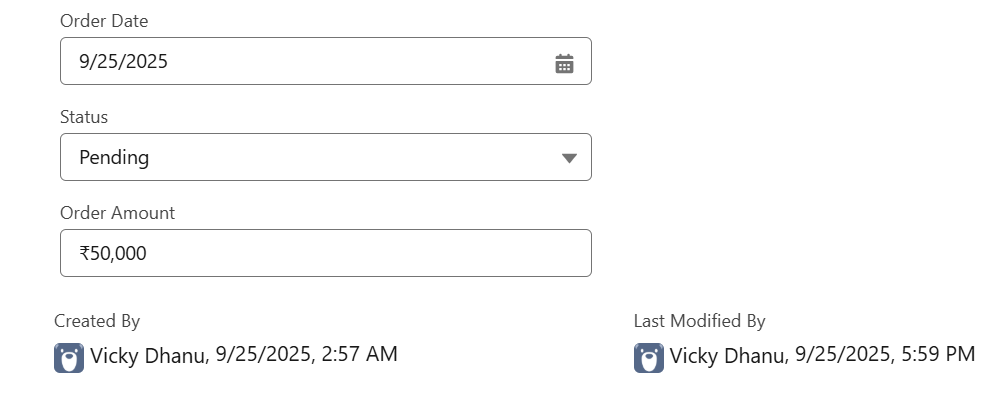
* For Inventory Object:



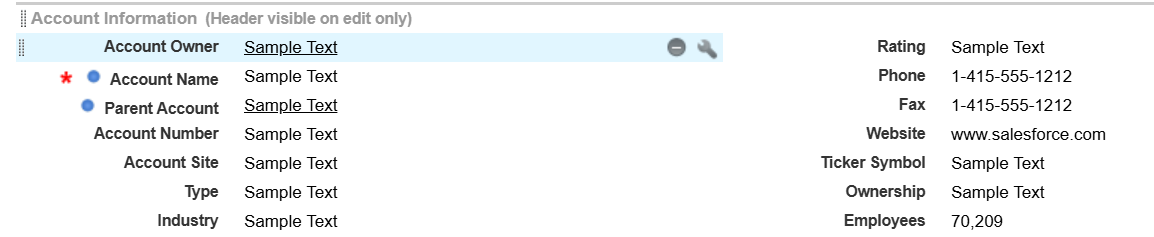
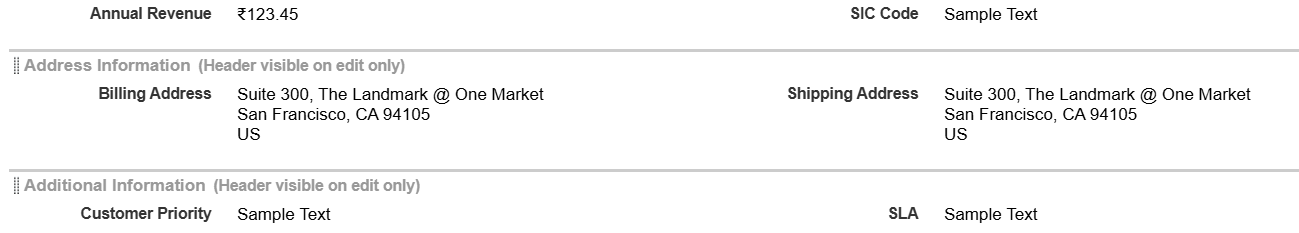
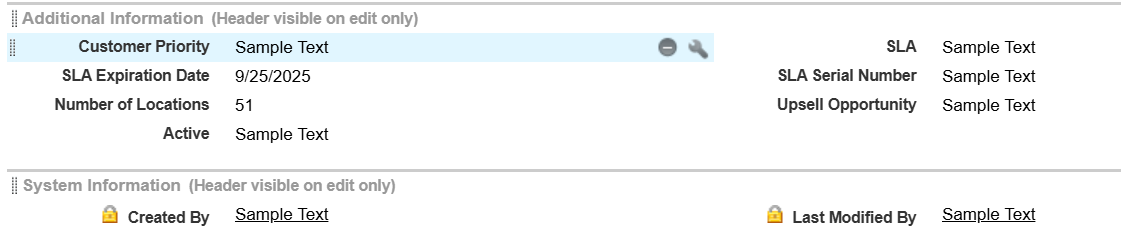
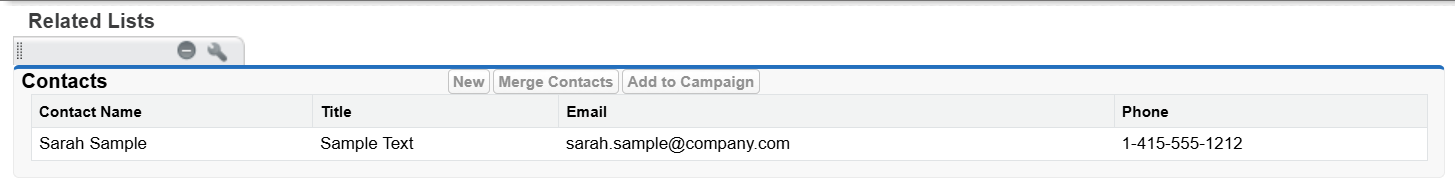
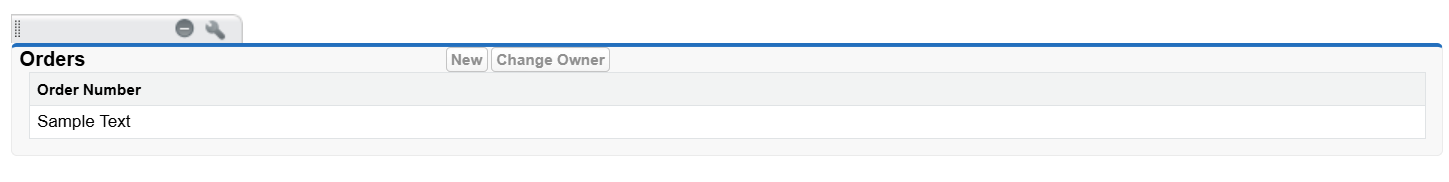
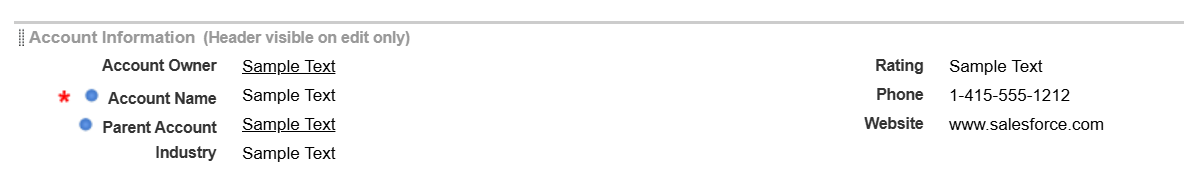


* For Order Object:



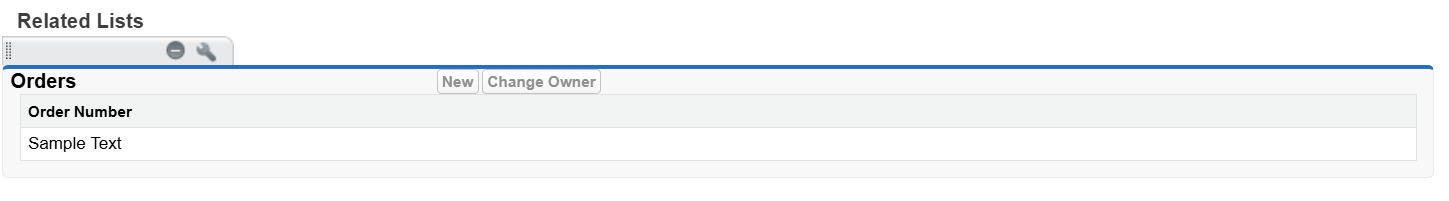


**Page Layouts:**

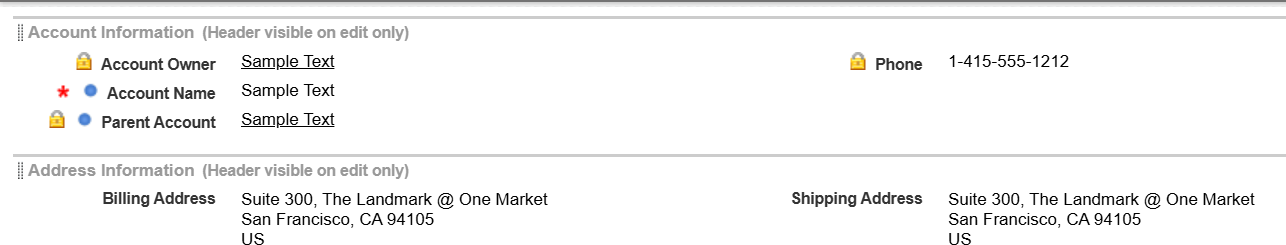
* Page Layouts control which fields and related lists are visible or editable for different profiles, ensuring users see only what’s relevant to their role.
* Steps to create the page layouts.
* Click on the gear icon and choose setup.
* Click on the object manager and in quick find search for the object that you need to create a layout.
* After going to the object go to the page layouts section.
* Next you clone the existing layout or create a new layout.
* In the page layout page, you can choose the choose the fields that you want to appear on that particular layout.
* You can also control the access by specifying the field as read – only or required.
* You can also add the related lists that you want to appear in the layout by dragging and dropping.
* You can save the layout as you wish and give a meaningful name so that you do not get confused while assigning the page layouts to the profiles.
* I have created Three page layouts for every object that I used in the project. They are   
   Inventory Manager Layout  
   Sales Rep Layout  
   Warehouse staff Layout
* I am attaching the screenshots for the page layouts of account object below. Inventory Manager Layout:  
     
    
    
    
    
    
    
  SalesRep Layout:  
  



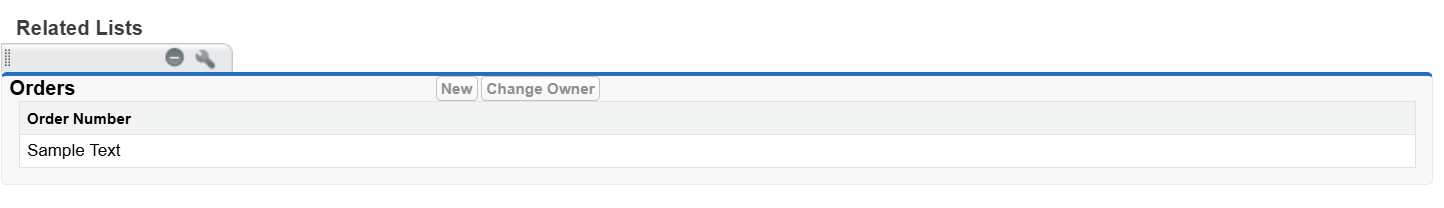




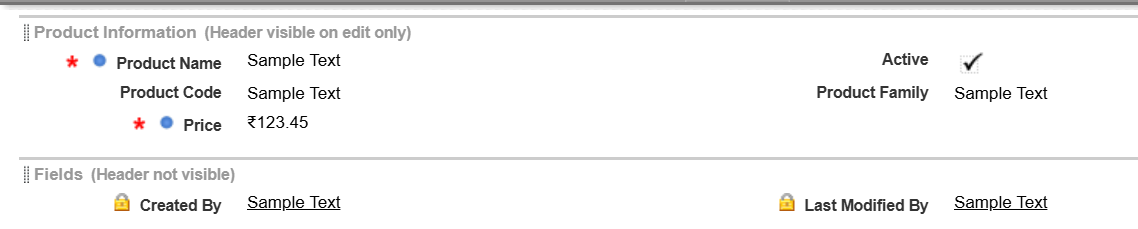
WarehouseStaff Layout:



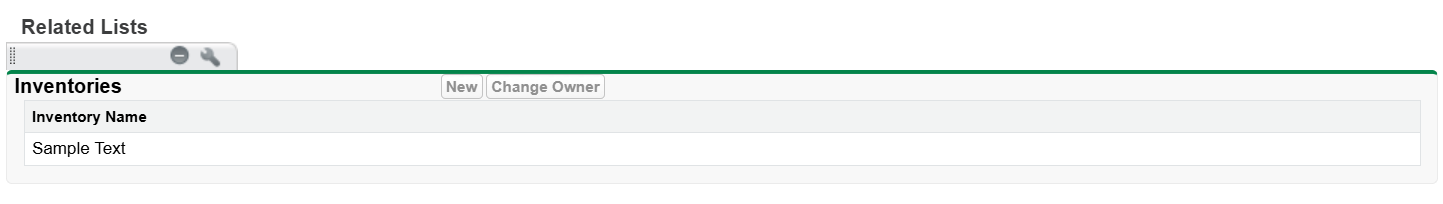
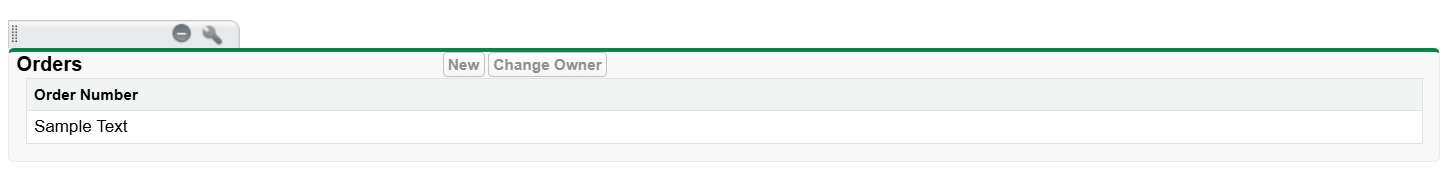




* I am attaching the screenshots for the page layouts of product object below.

Inventory Manager Layout:  


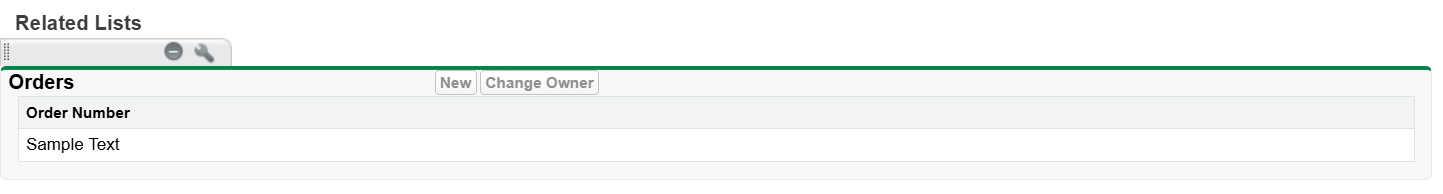


SalesRep Layout:



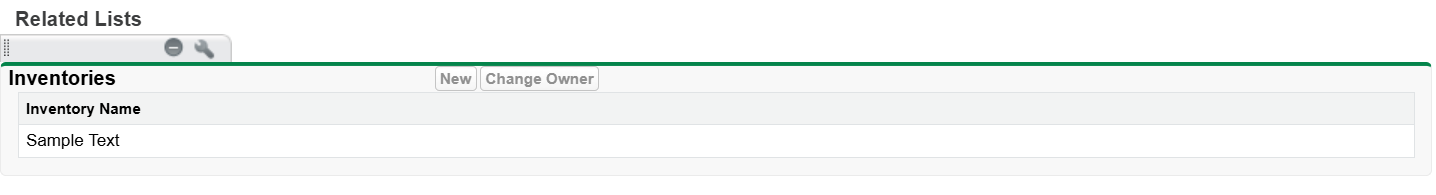




WarehouseStaff Layout:

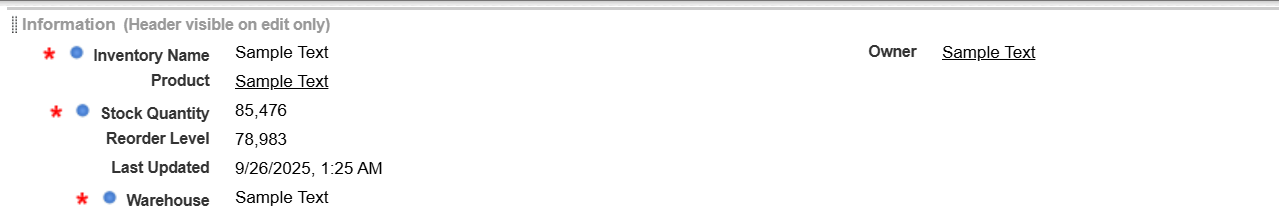




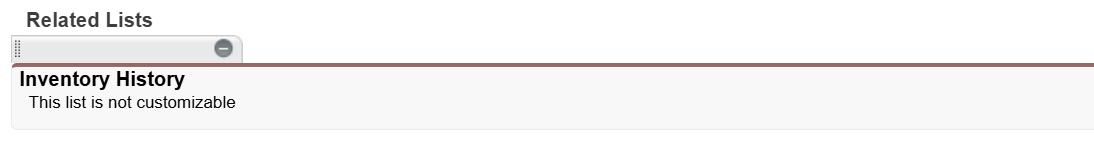


* I am attaching the screenshots for the page layouts of Inventory object below.

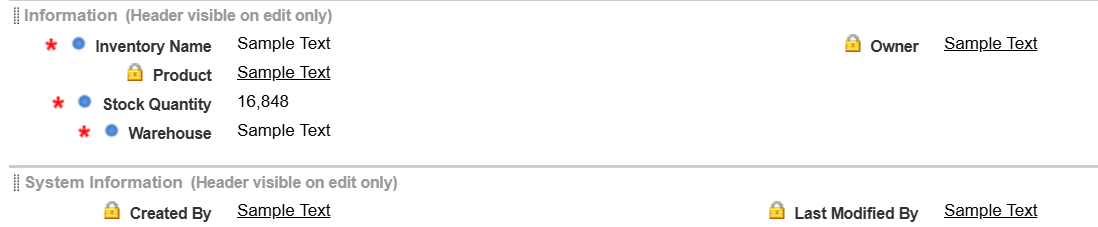
Inventory Manager Layout:

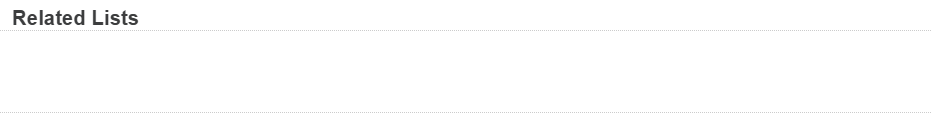




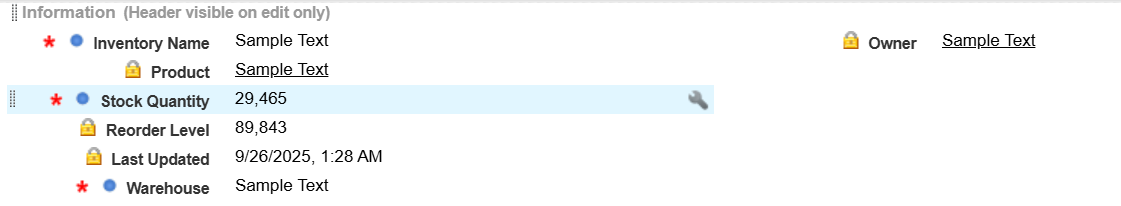


SalesRep Layout:

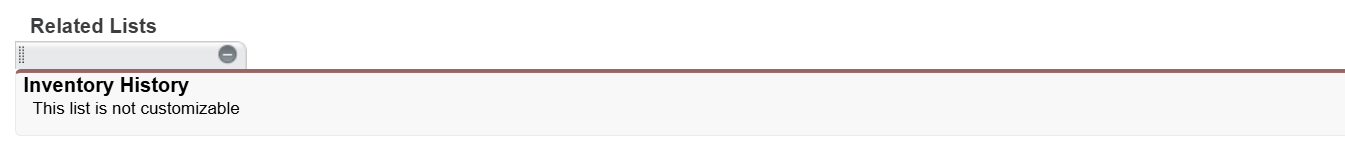




WarehouseStaff Layout:

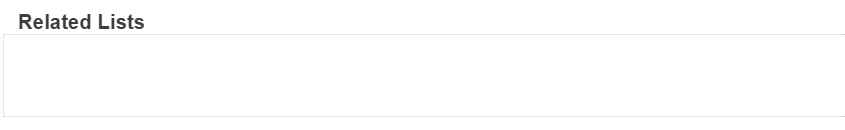






* I am attaching the screenshots for the page layouts of Order object below. Inventory Manager Layout:  
  

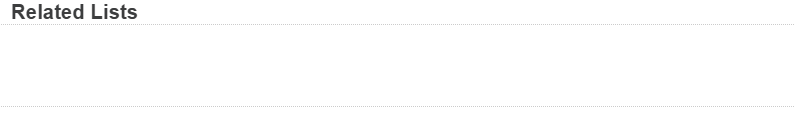




SalesRep Layout:



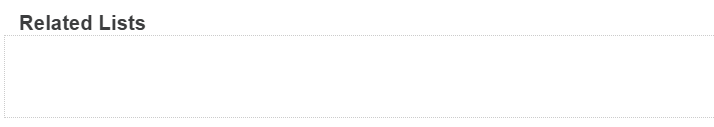




WarehouseStaff Layout:

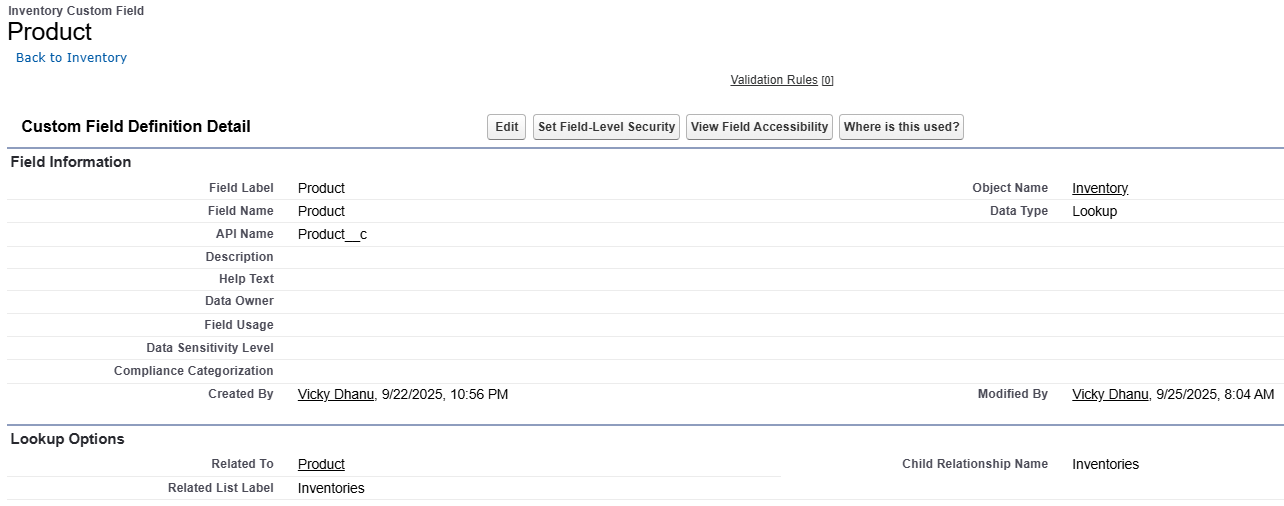
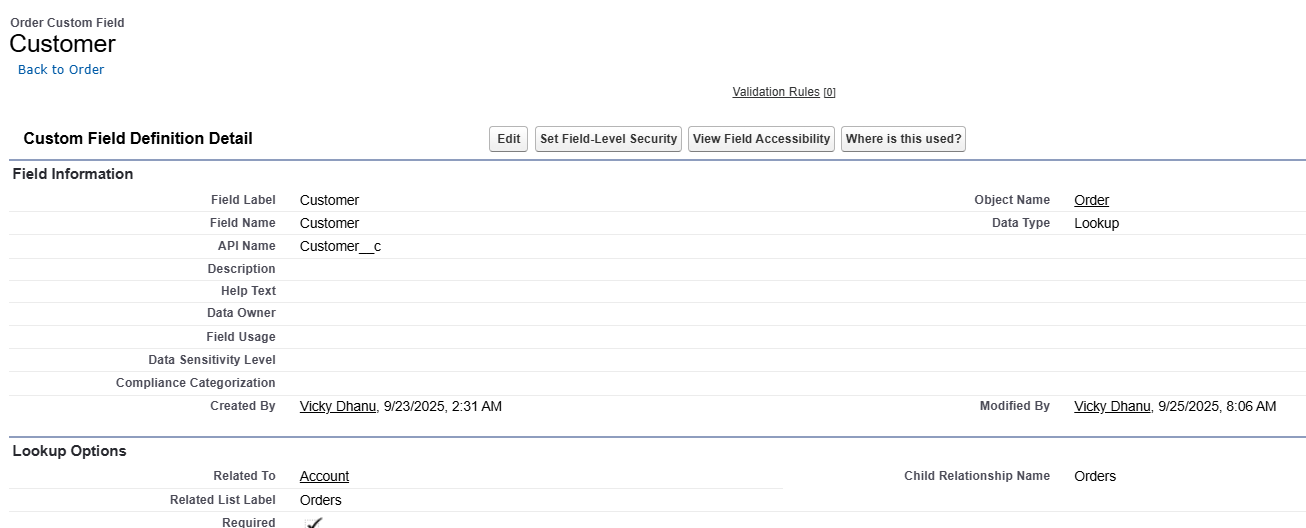
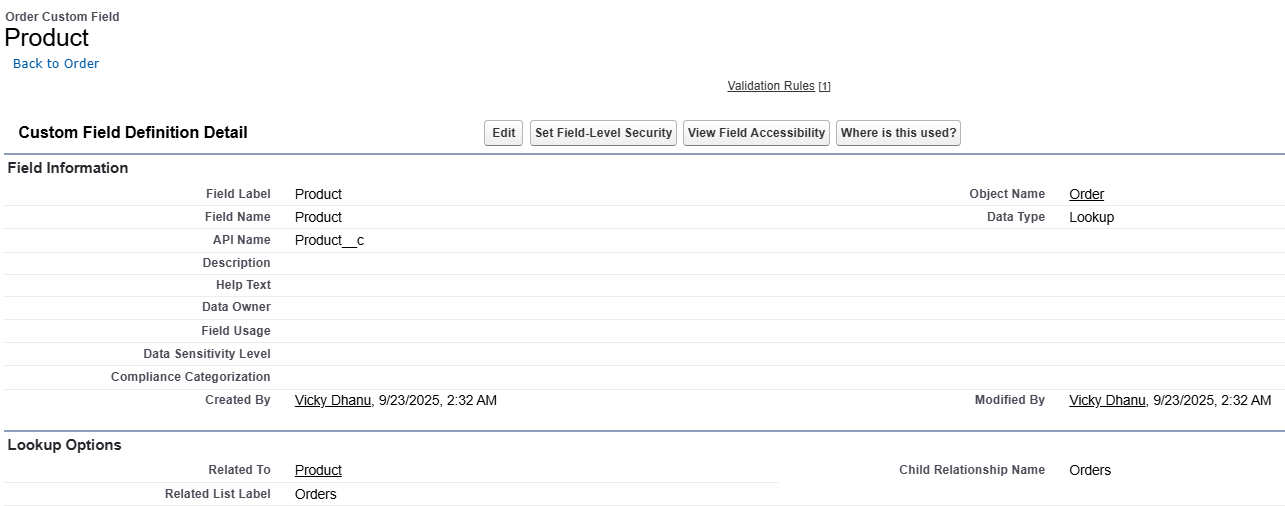






* The above the are the 12 layouts that I have created as part of the project.
* The lock icon in the layout indicates that those fields are only readable.
* The red asterisk in the layout indicates that those fields are compulsory.

**Lookup Relationship:**

* A lookup relationship in Salesforce is a loosely coupled relationship between two objects. It allows one object to reference another without strict dependency.
* In Inventory object I have created a lookup relationship with product.
* In Order object I have created a lookup relationship with the account and Product object.
* I have attached the screenshots below
* This is the lookup relation of inventory object with product.  
  
* This is the lookup relation of order object with account.  
  
* This is the lookup relation of order with product.  
  

Phase 4: Process Automation

**Validation Rules**Validation rules in Salesforce are used to enforce data quality by preventing users from saving invalid or incorrect records. They check the values entered into fields and display error messages when criteria are not met.

Key Benefits:

* Improves data accuracy.
* Prevents incomplete or illogical entries.
* Guides users with clear error messages.

Steps to create validation rule  
 **Navigate to Object Manager**

* Go to **Setup → Object Manager**.
* Select the object (e.g., Order\_\_c).

**Open Validation Rules**

* In the left panel, click **Validation Rules**.
* Click **New**.

**Enter Rule Details**

* **Rule Name**: Provide a clear name (e.g., Order\_Quantity\_Positive).
* **Description**: (Optional) Add notes for clarity.

**Define the Formula**

* Write the condition when the record should be invalid (e.g., Quantity\_\_c <= 0).
* Test syntax to confirm.

**Set Error Message & Location**

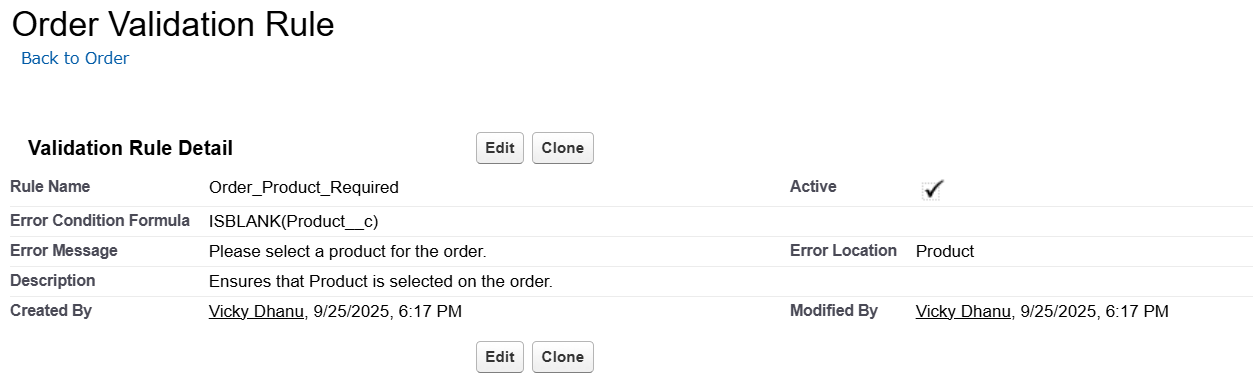
* Add a **user-friendly error message** (e.g., *“Order quantity must be greater than zero.”*).
* Choose where to display it (Field level or Top of Page).

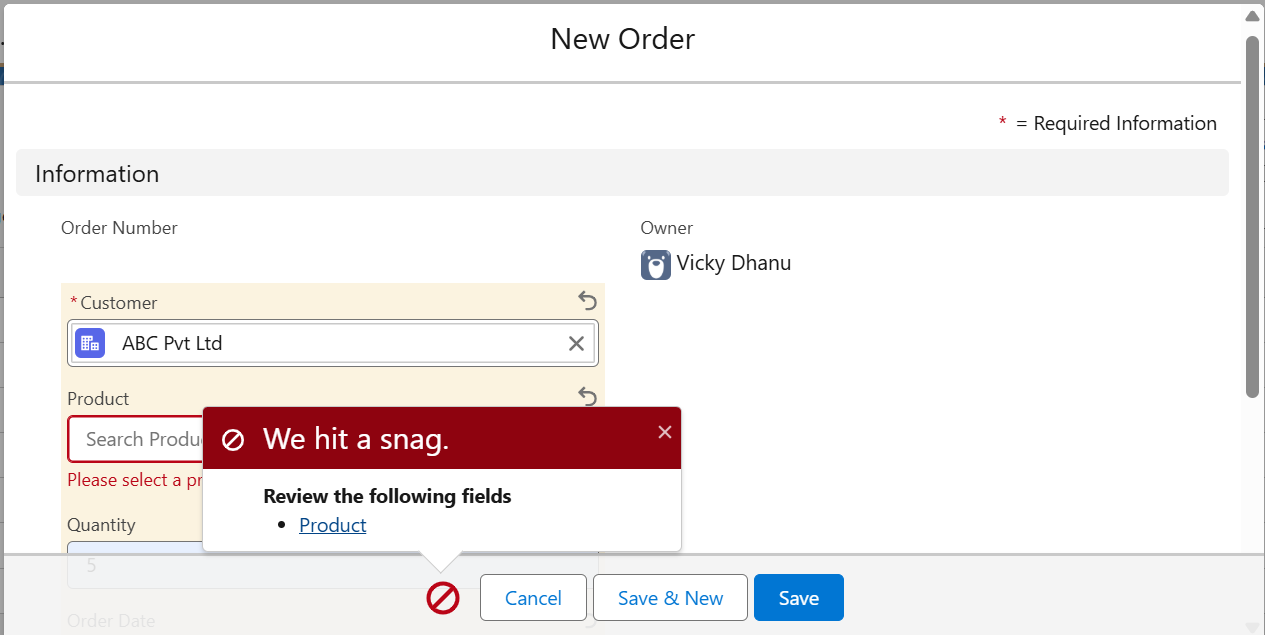
**Save & Test**

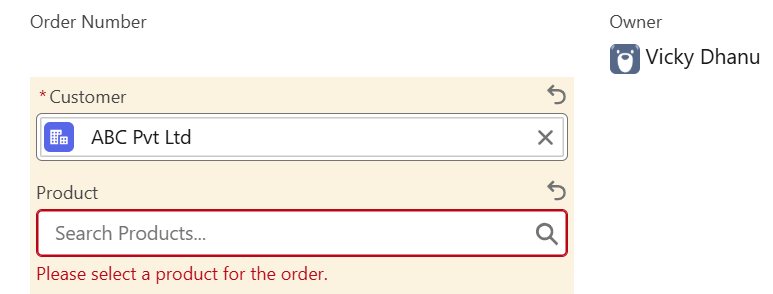
* Save the rule.
* Try creating or updating a record to ensure the validation works.

I have created two validation rules in my project for orders object.

First One is that the product should be selected compulsorily for every object.

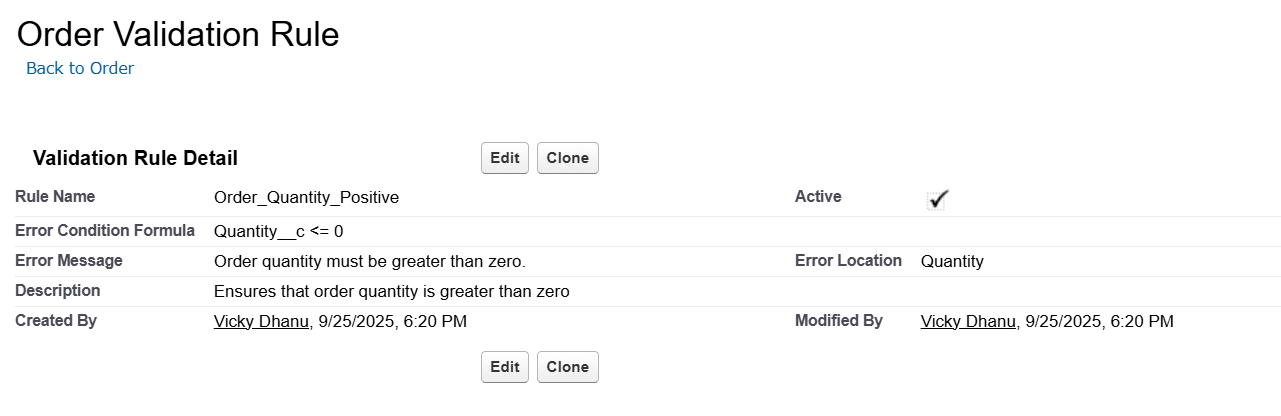
Demonstration of the first validation rule

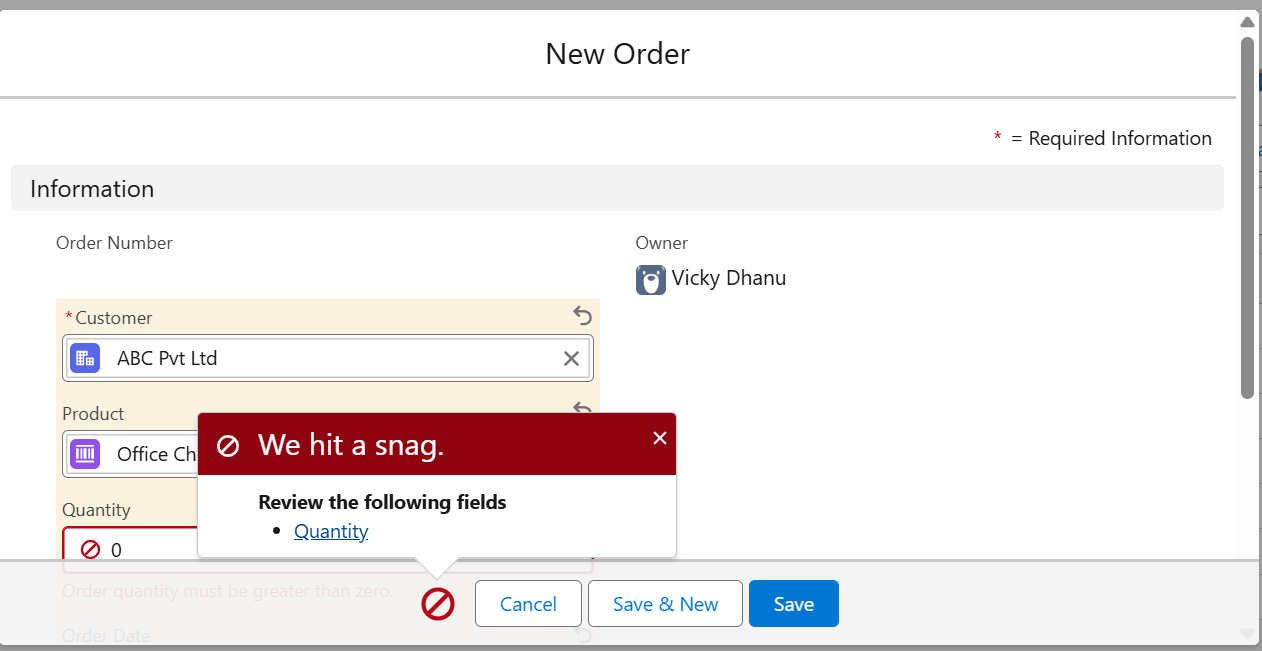


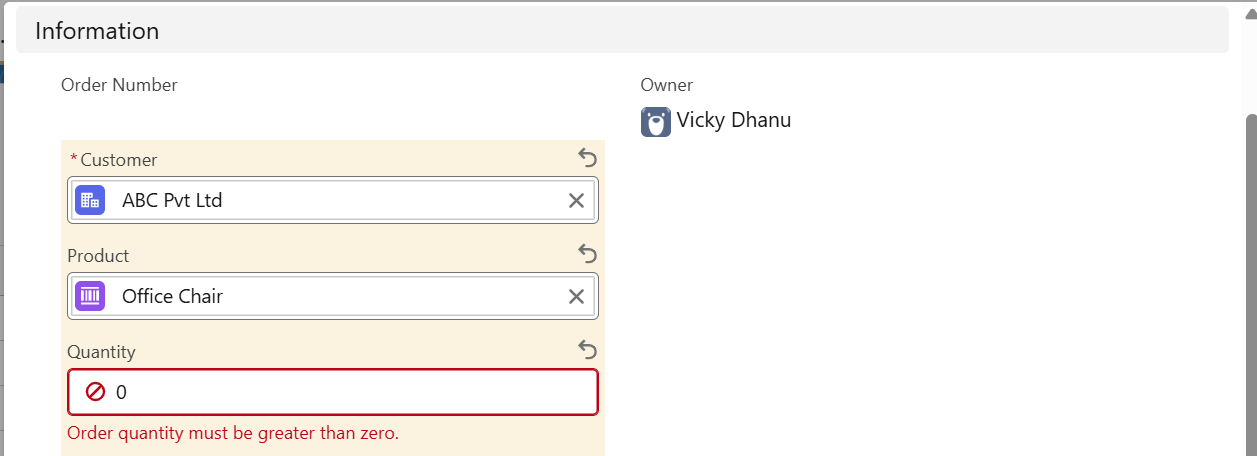


I have tried to create a new order without a product but the validation prevented the creation of order.

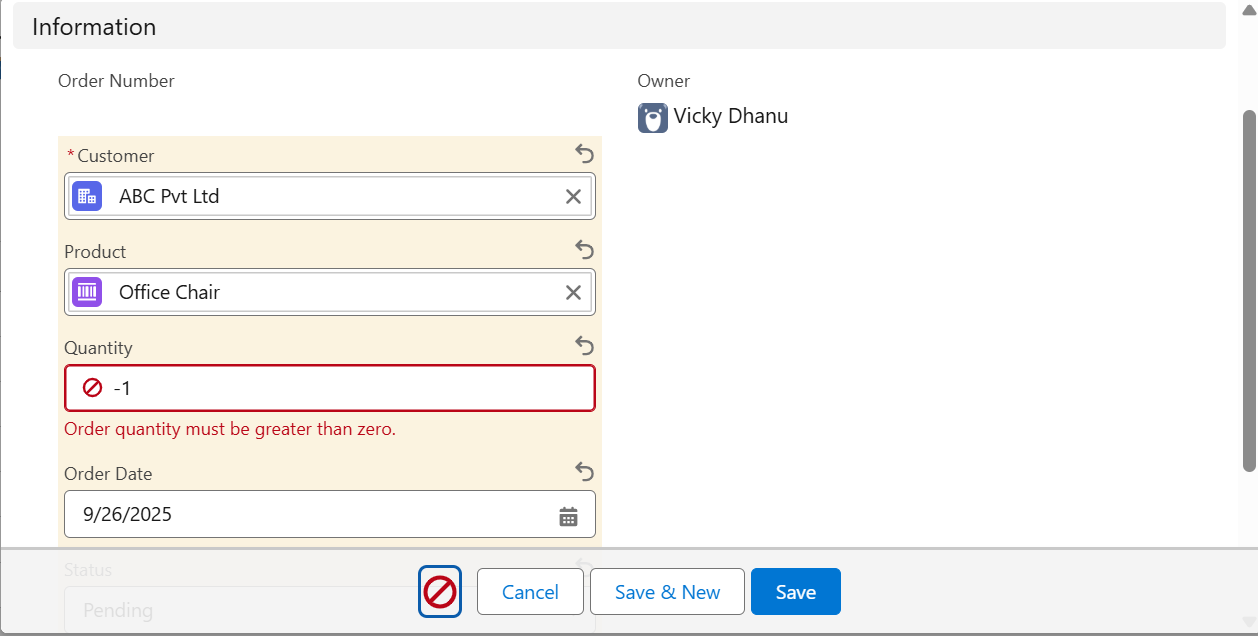
Second One is that the product Quantity should be greater than 0.

****

Demonstration of second validation rule  




I have tried to create a new order with a product and quantity is set to zero the validation prevented the creation of order.



I have tried to create a new order with a product and quantity is set to less than zero the validation rule prevented the creation of order.

**Flow Builder**

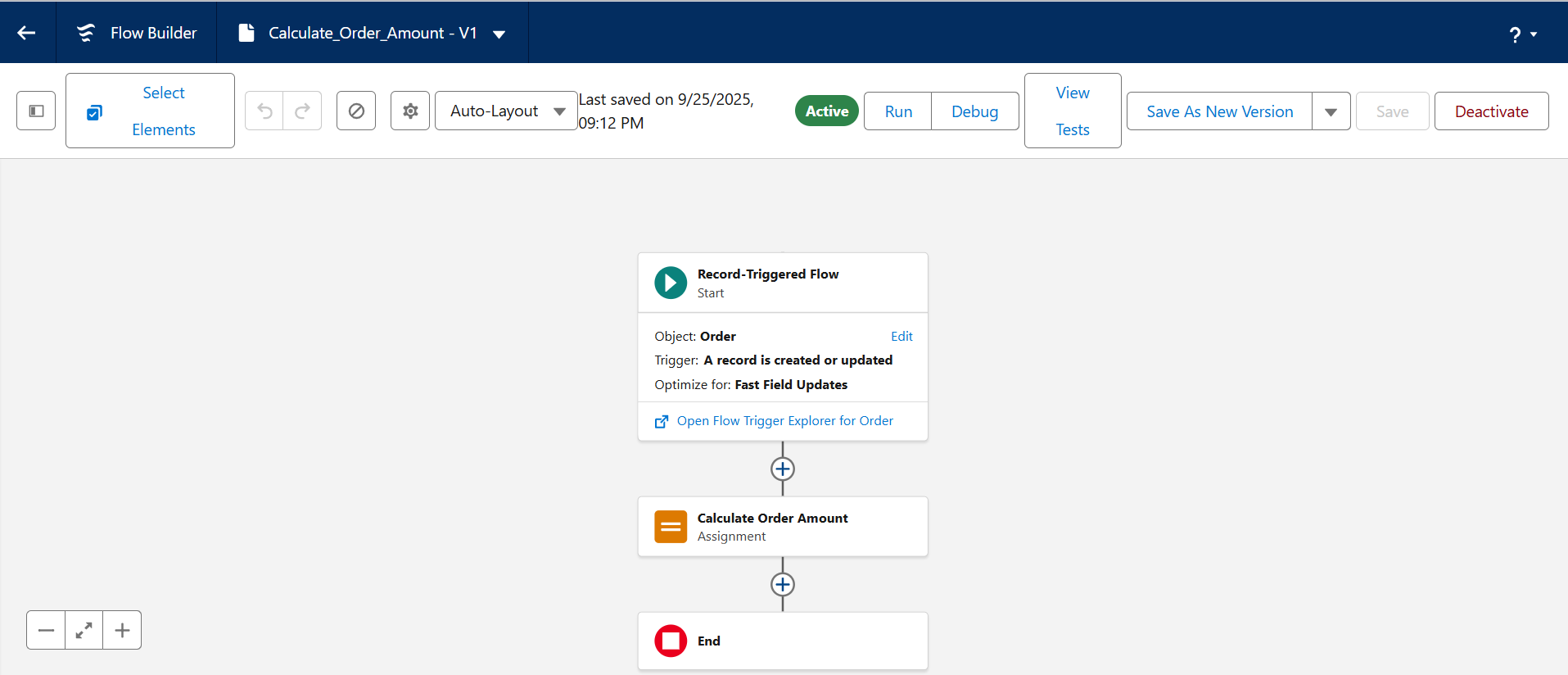
Flow Builder is a point-and-click automation tool in Salesforce that allows admins to build powerful business processes without code. It can perform actions like creating or updating records, sending emails, posting notifications, or even calling Apex code.

Flows can be:

* Record-Triggered Flows → Run automatically when records are created/updated/deleted.
* Screen Flows → Provide interactive screens for users to input data.
* Scheduled/Auto-Launched Flows → Run on a schedule or background logic.

I have created three record triggered flows in the project.

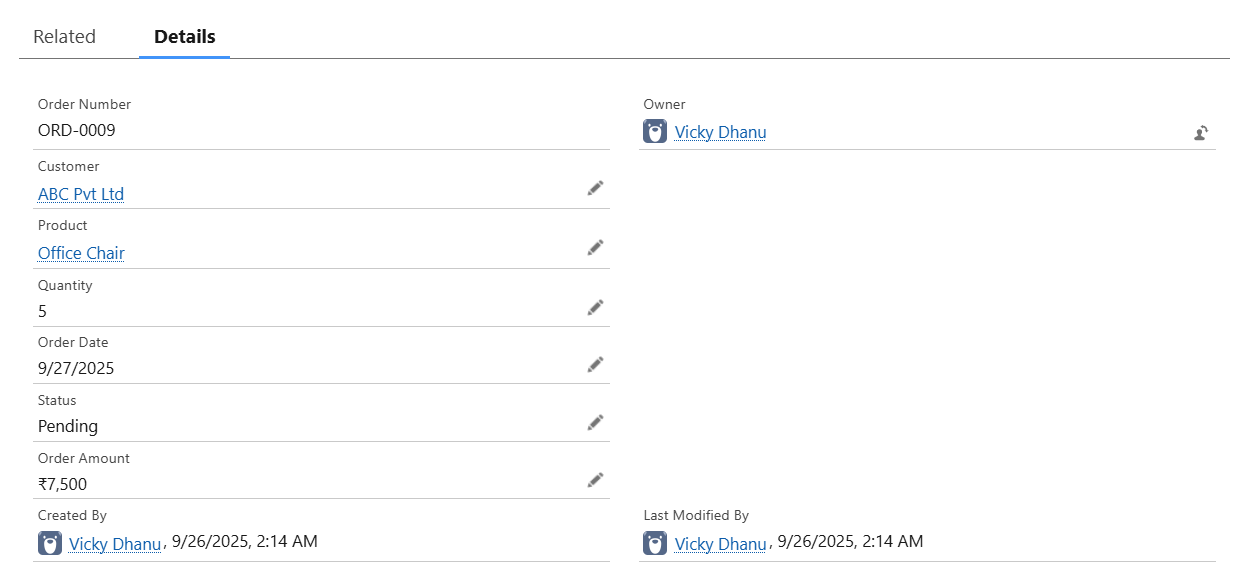
First one is to calculate the total order amount whenever I create an order I give the quantity but I do not set total price this flow calculates the price and sets the value into that field.



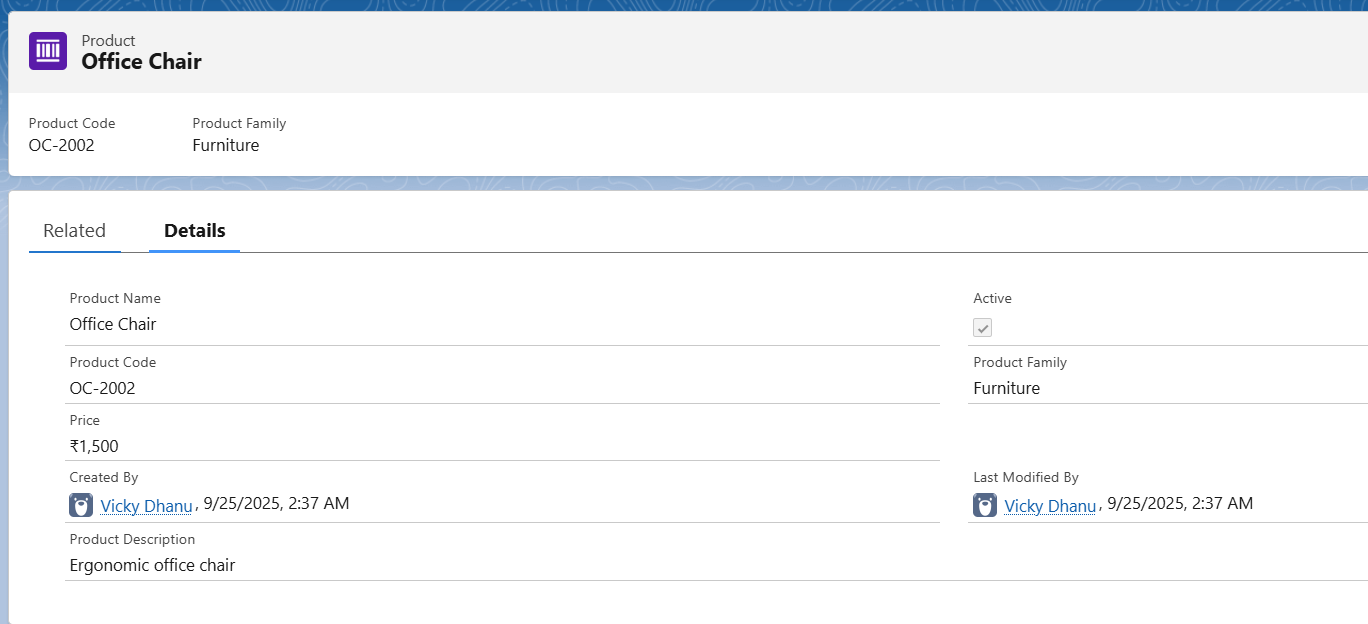
Demonstration of the flow  
The below image is the order at the time of creation



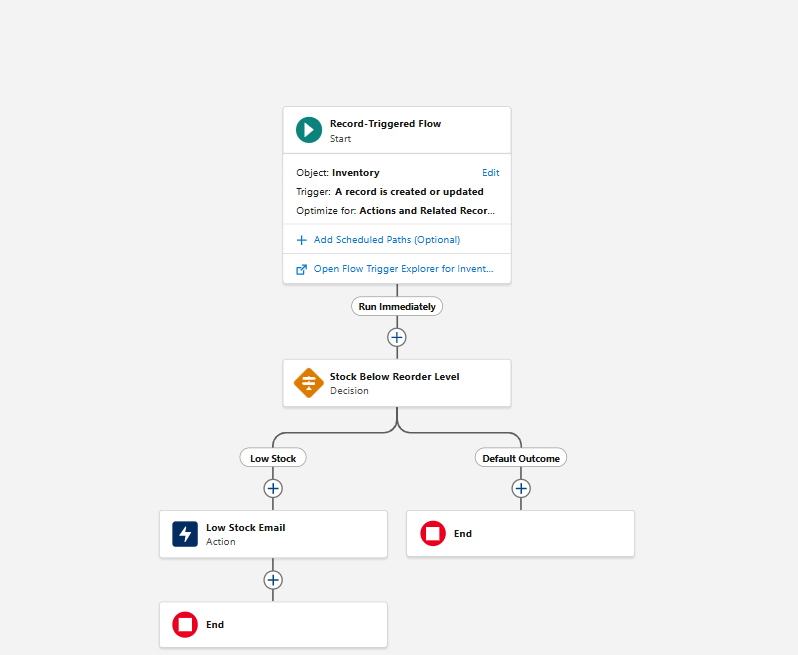
The below image after the creation of order

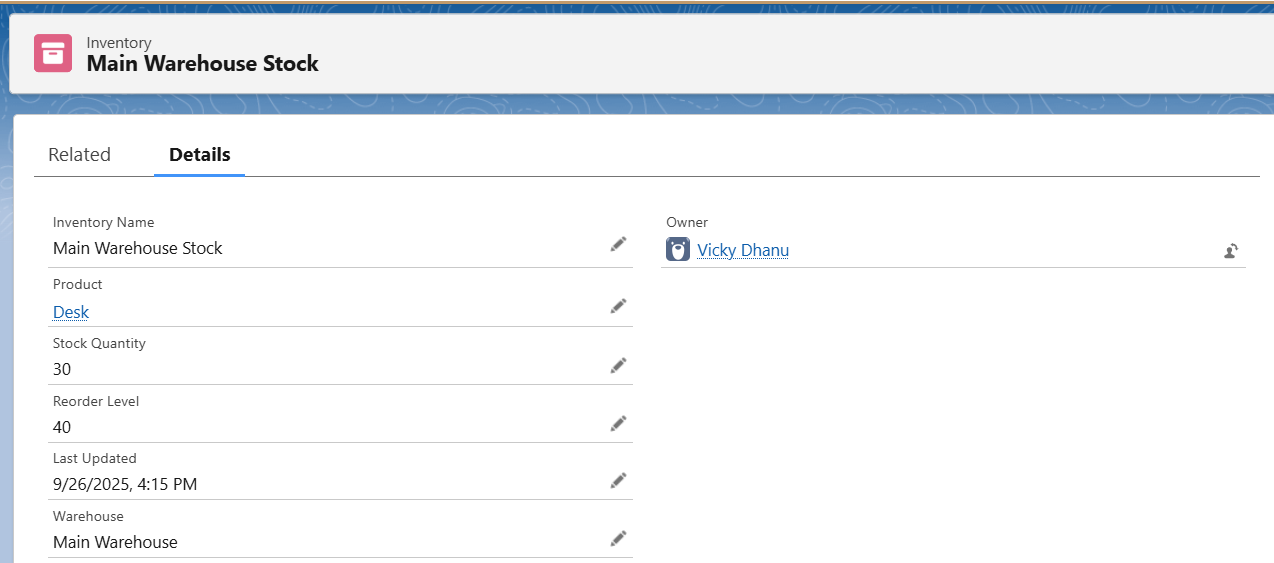


The order amount field is empty before and now it is populated. The below is the image of the product record.

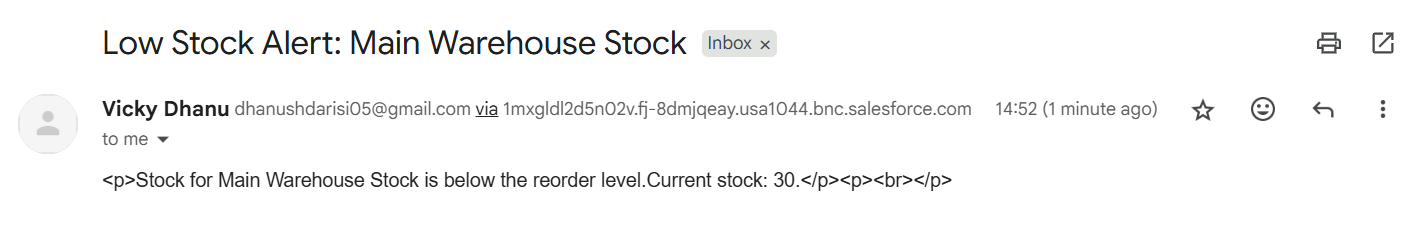


The Second one is the low stock alert flow whenever the stock quantity goes below the reorder level an email alert is sent to the inventory manager.

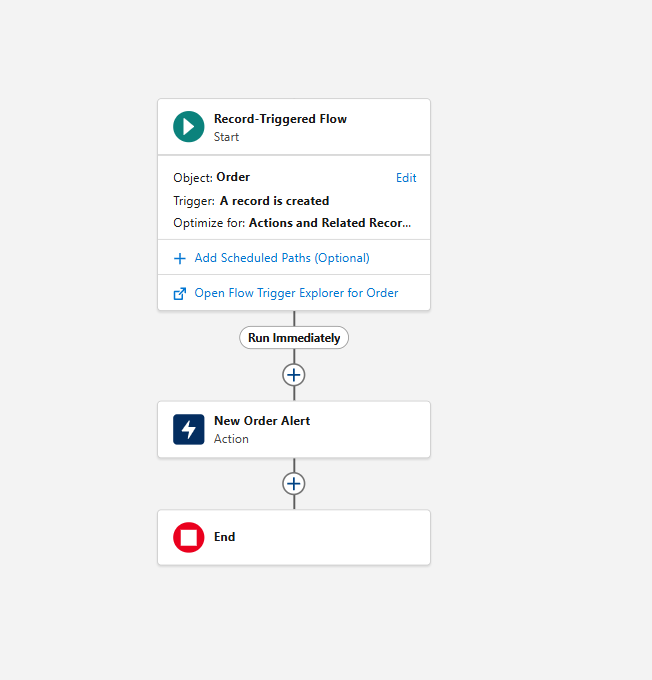


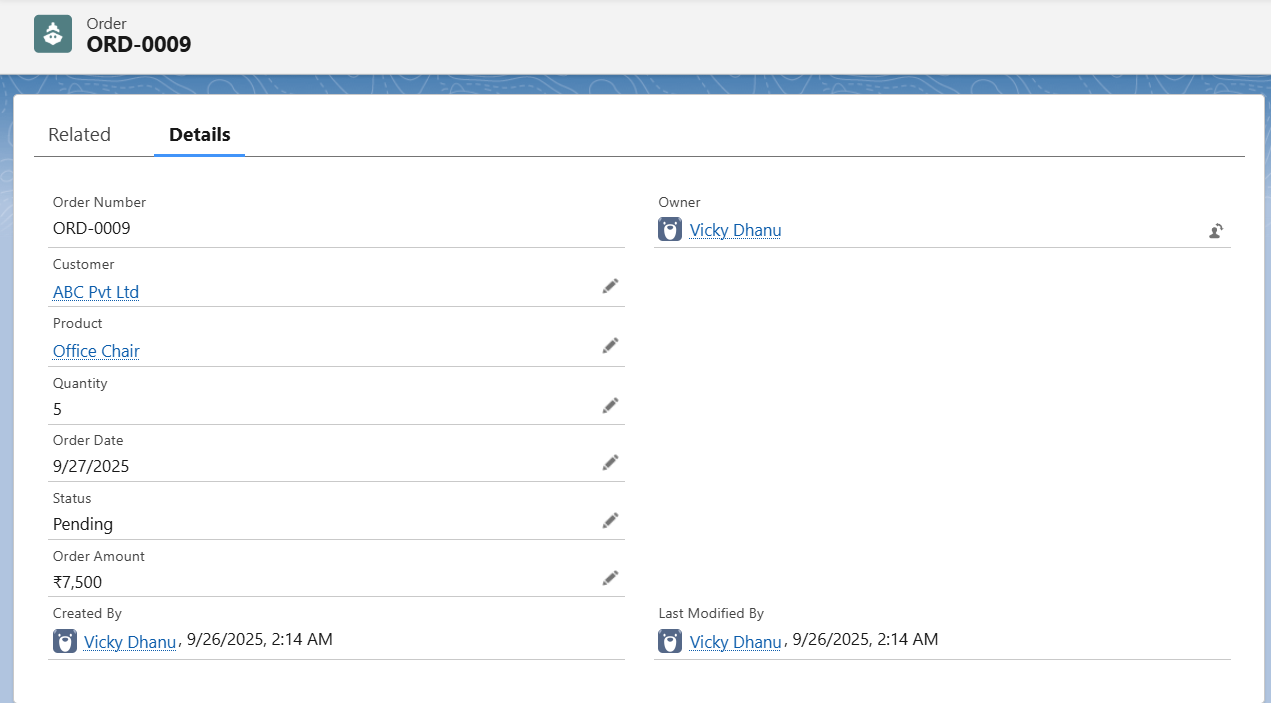
Demonstration of flow  


In the above picture the stock quantity is less than the reorder level. So Inventory manager got a mail. The below is the image of the mail he got.

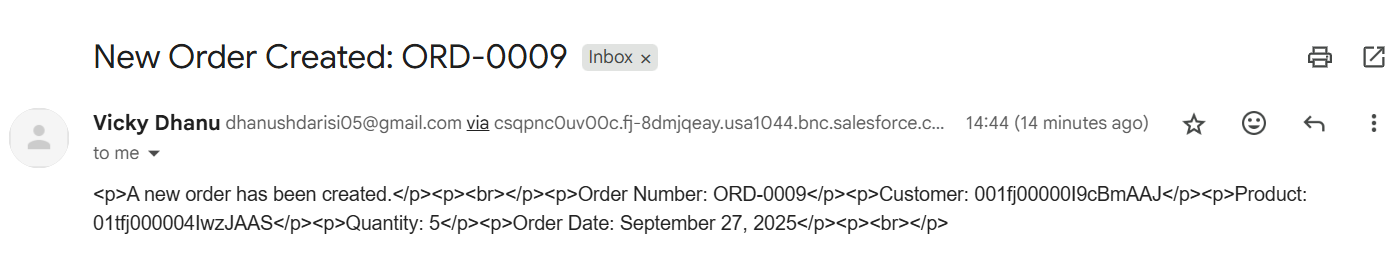


The third one is the order creation alert flow whenever an order is created Inventory Manager gets a email notification.



Demonstration of the flow  
The below is the image of the order that I have created  


Immediately after the order is created Inventory manager got an email regarding the order creation.



Phase 5: Apex Programming

**Apex Triggers**Apex Triggers are pieces of code that automatically execute before or after database operations (like insert, update, delete) on a record. Triggers allow developers to enforce business rules, update related records, and automate complex logic that cannot be handled by declarative tools.

In this project, an Apex Trigger was created on the Order object to automatically reduce stock levels in the Inventory whenever an order is approved, ensuring real-time inventory management.  
  
The code for the apex trigger is below  
trigger UpdateInventoryOnOrder on Order\_\_c (after insert, after update) {

for (Order\_\_c o : Trigger.new) {

Order\_\_c oldOrder = Trigger.isUpdate ? Trigger.oldMap.get(o.Id) : null;

// Reduce stock only when status changes to Approved

Boolean becameApproved = (

o.Status\_\_c == 'Approved' &&

(oldOrder == null || oldOrder.Status\_\_c != 'Approved')

);

if (becameApproved && o.Product\_\_c != null && o.Quantity\_\_c != null) {

// Find matching inventory

Inventory\_\_c inv = [

SELECT Id, Stock\_Quantity\_\_c

FROM Inventory\_\_c

WHERE Product\_\_c = :o.Product\_\_c

LIMIT 1

];

// Reduce stock

inv.Stock\_Quantity\_\_c -= o.Quantity\_\_c;

if (inv.Stock\_Quantity\_\_c < 0) {

inv.Stock\_Quantity\_\_c = 0; // avoid negative stock

}

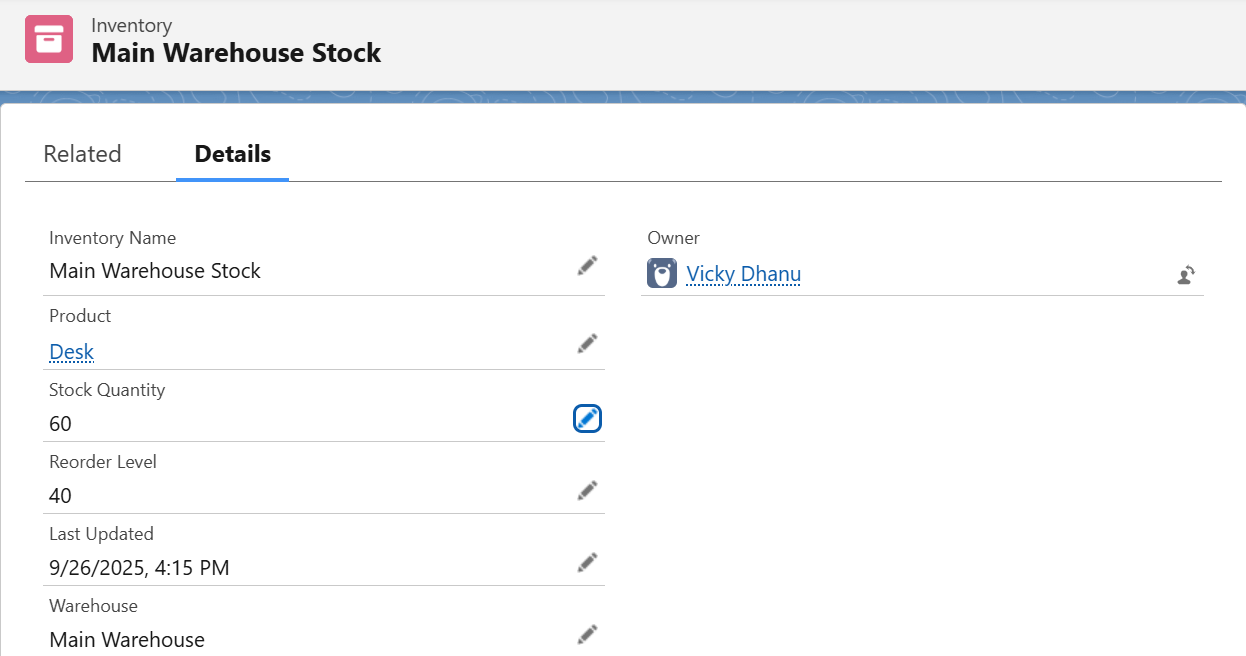
update inv;

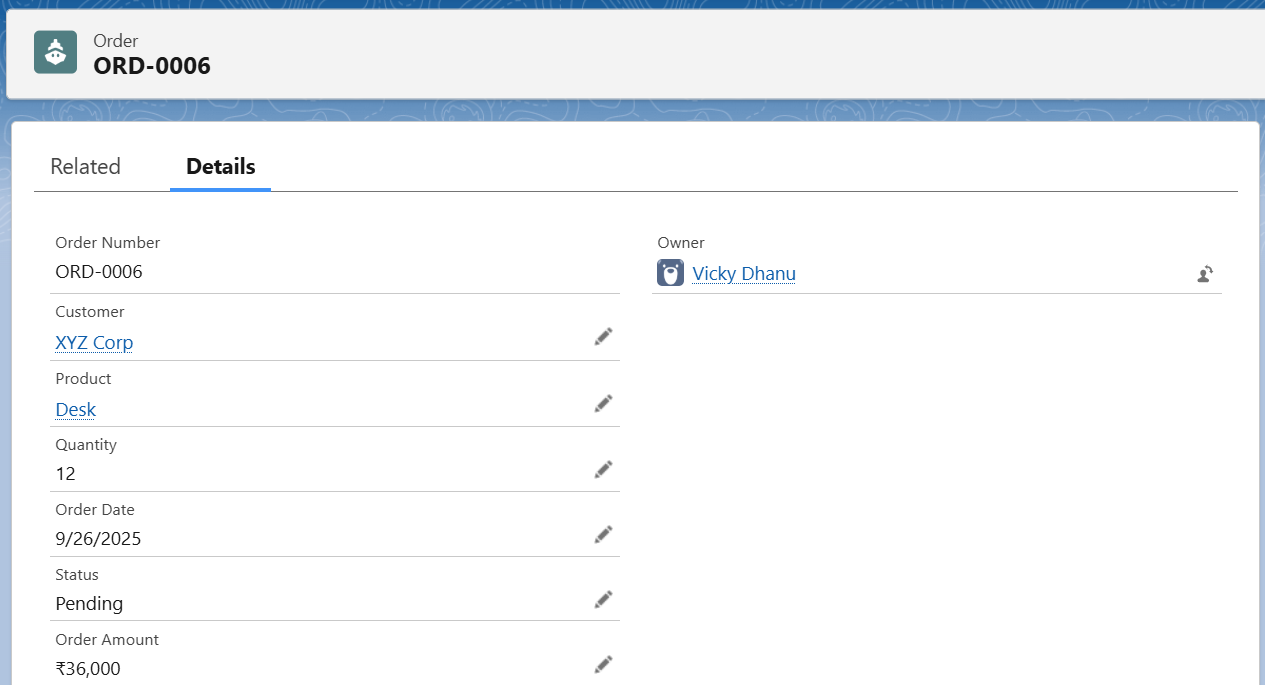
}

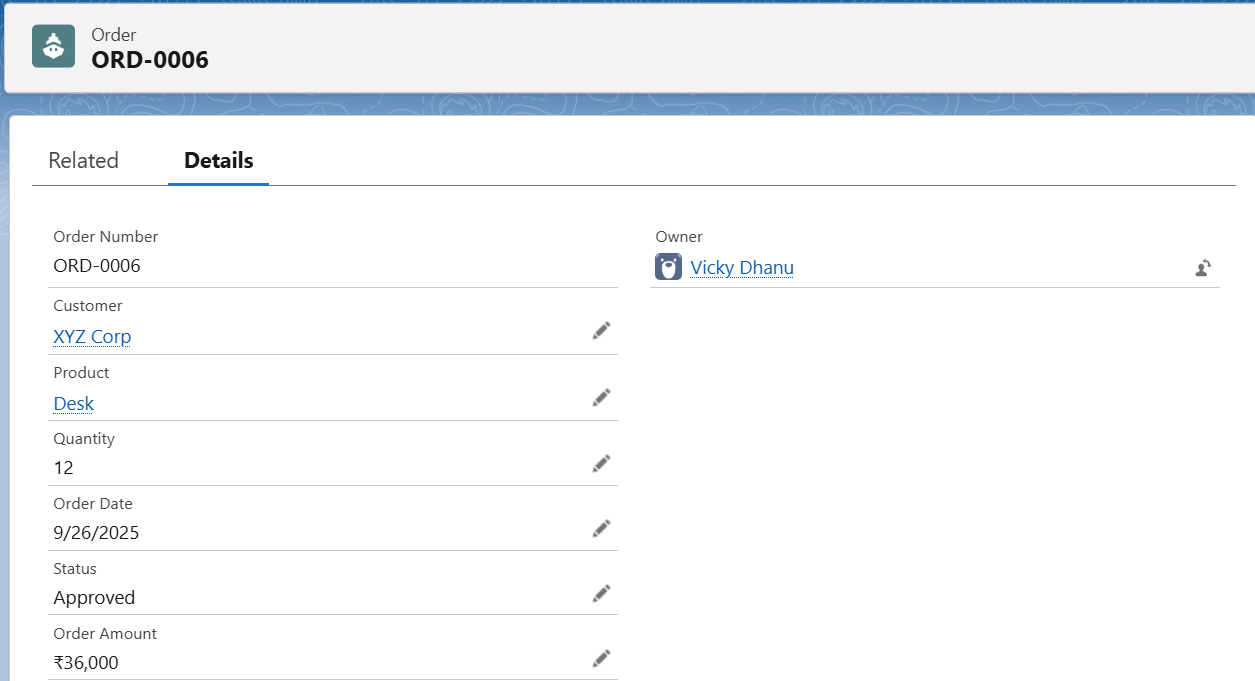
}

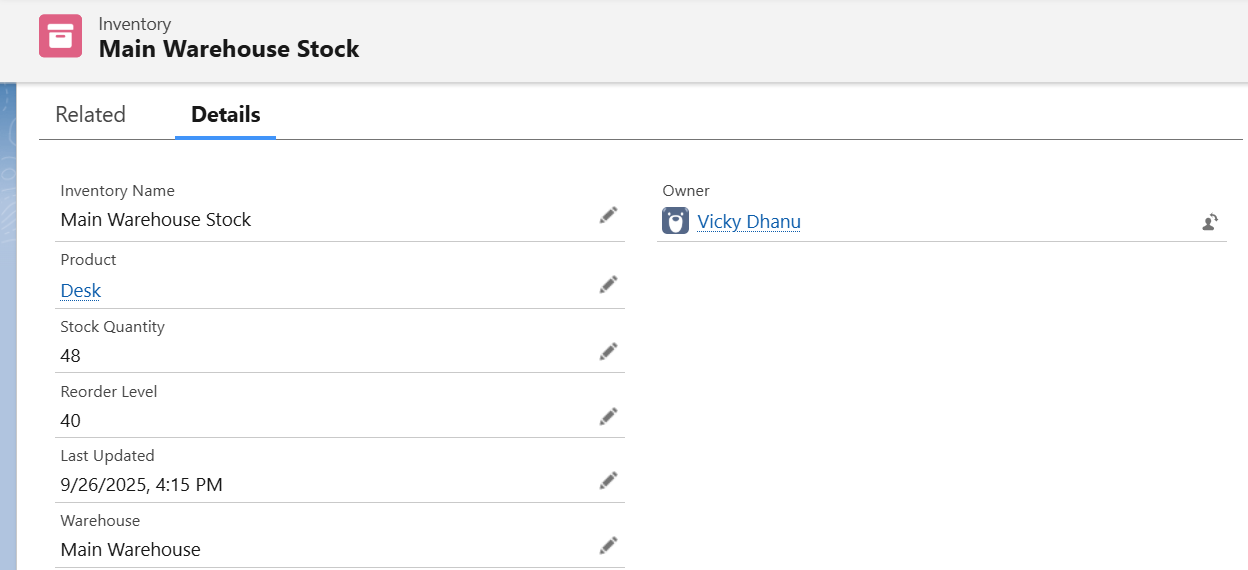
}  
  
The UpdateInventoryOnOrder trigger runs after insert and after update on the Order\_\_c object. Its purpose is to ensure inventory stock is reduced only when an order’s status changes to Approved.

* It checks if the order status became Approved (newly approved orders).
* It finds the related inventory record for the selected product.
* It reduces the Stock Quantity by the ordered quantity.
* It ensures stock never goes below zero.

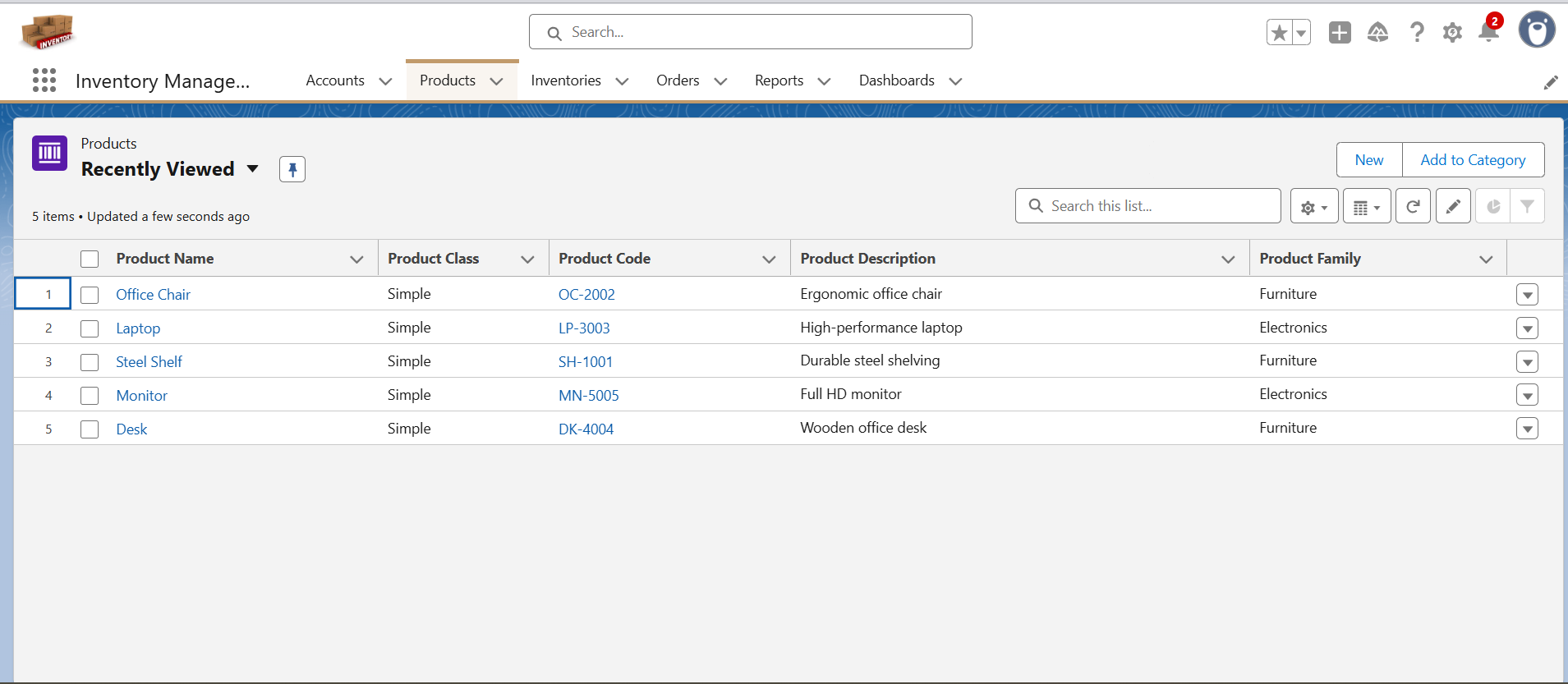
Demonstration of the trigger  
 The below is the image of the inventory record when the order is in pending state.  
 

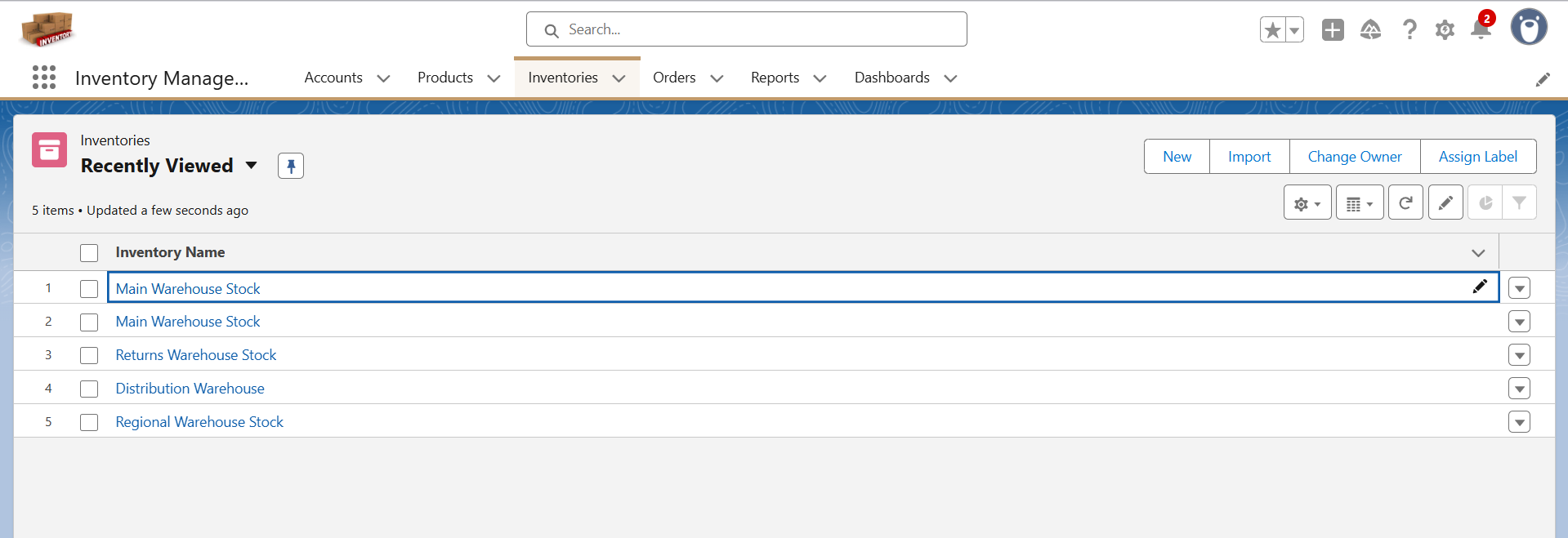
The below is the order image in pending state  


The order status is changed to approved in the below image  


The inventory record stock quantity is updated   


Phase 6: User Interface Development  
An app named Inventory Management System was created using App Manager to provide easy navigation between Accounts, Products, Orders, and Inventory objects. Lightning App Builder was not used, as standard page layouts were sufficient for this project.

The below is the screenshot of the app that I created  






Phase 7: Integration & External Access

As of now, there is no requirement to implement the features in this phase. In the future, some of these capabilities may be explored and implemented as needed.

Phase 8: Data Management & Deployment

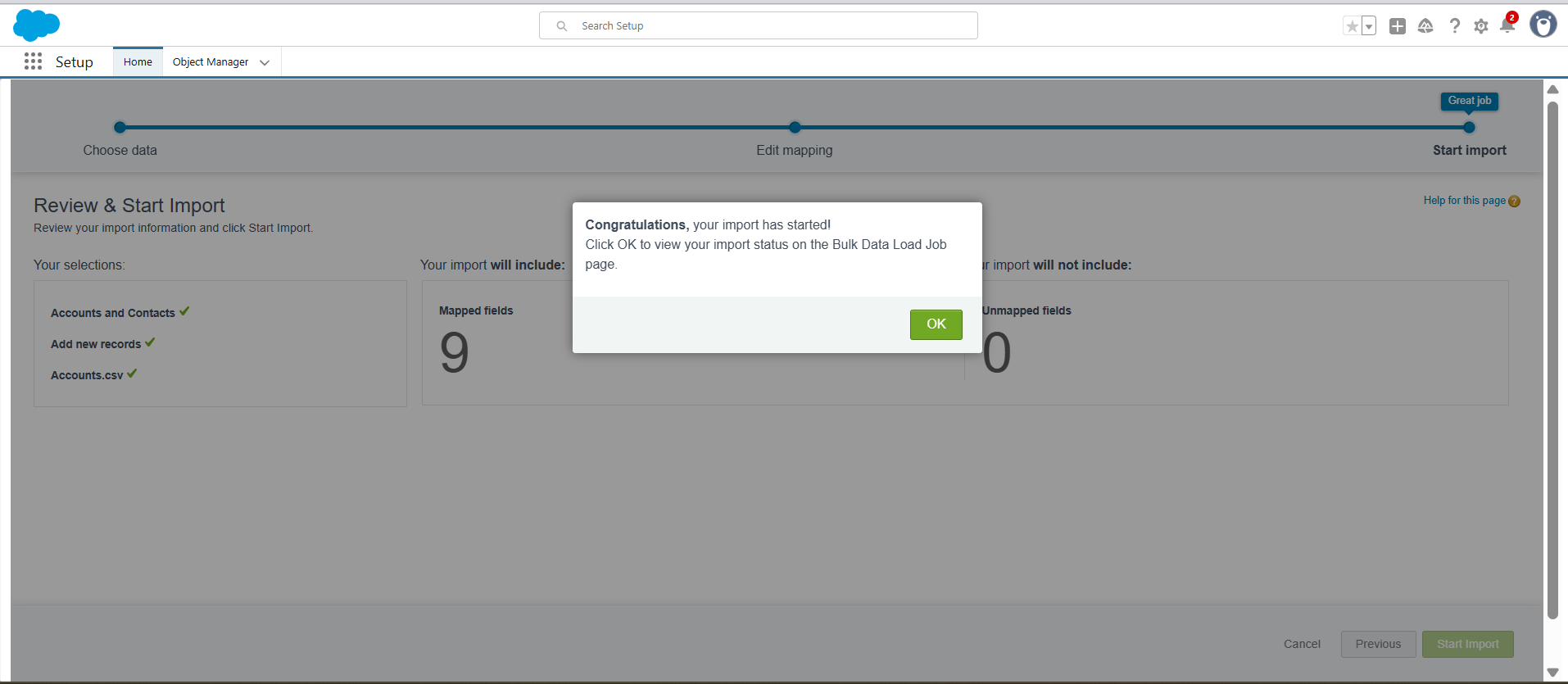
**Data Import Wizard:**  
The Salesforce Data Import Wizard is a built-in, point-and-click tool for importing data into Salesforce. It allows users to insert, update, or upsert records for both standard and custom objects without writing code.

**Key Features:**

* Supports standard objects (Accounts, Contacts, Leads) and custom objects (e.g., Inventory, Orders).
* Can insert new records, update existing records, or upsert using Salesforce IDs or External IDs.
* Provides field mapping, letting users map CSV columns to Salesforce fields.
* Automatically assigns imported records to the current logged-in user if OwnerId is not provided.
* Generates error reports for failed records, which can be downloaded for correction.

**Advantages:**

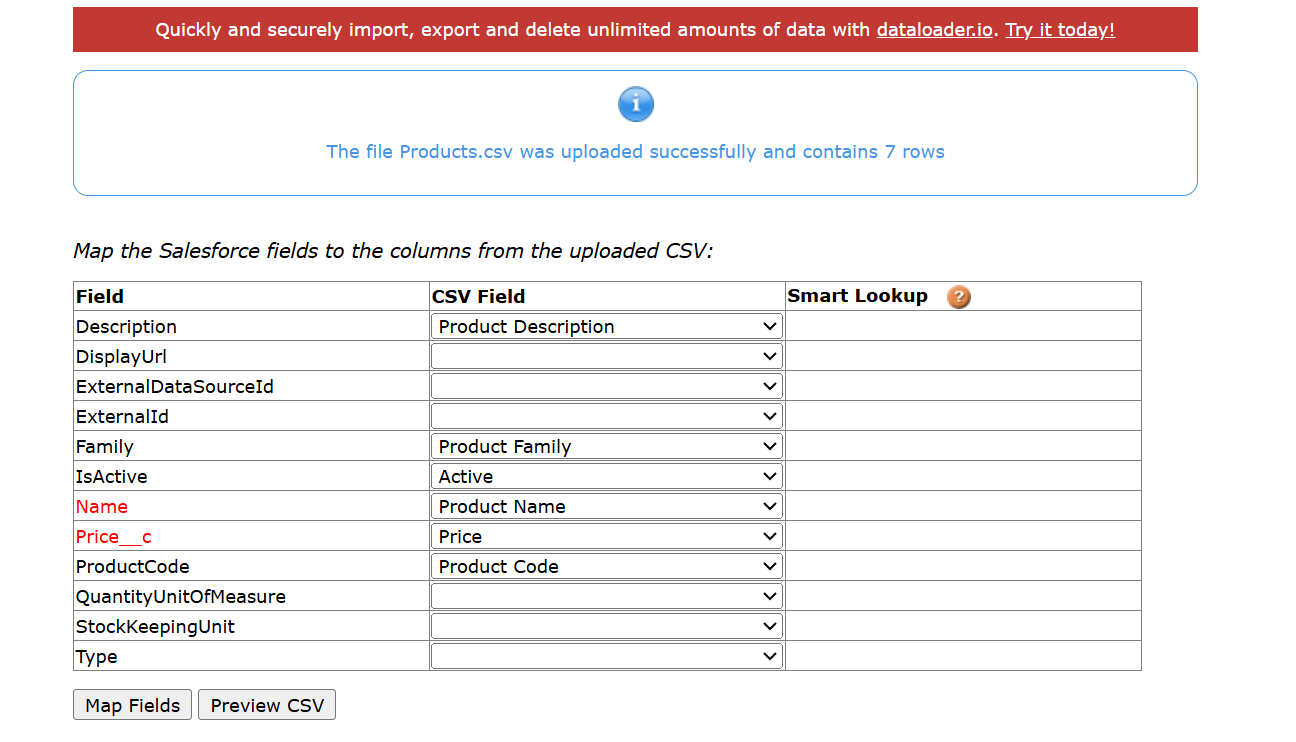
* User-friendly and requires no coding skills.
* Ideal for small to medium datasets (up to 50,000 records).
* Handles related records via External IDs.
* Quick and efficient for testing or demo purposes.

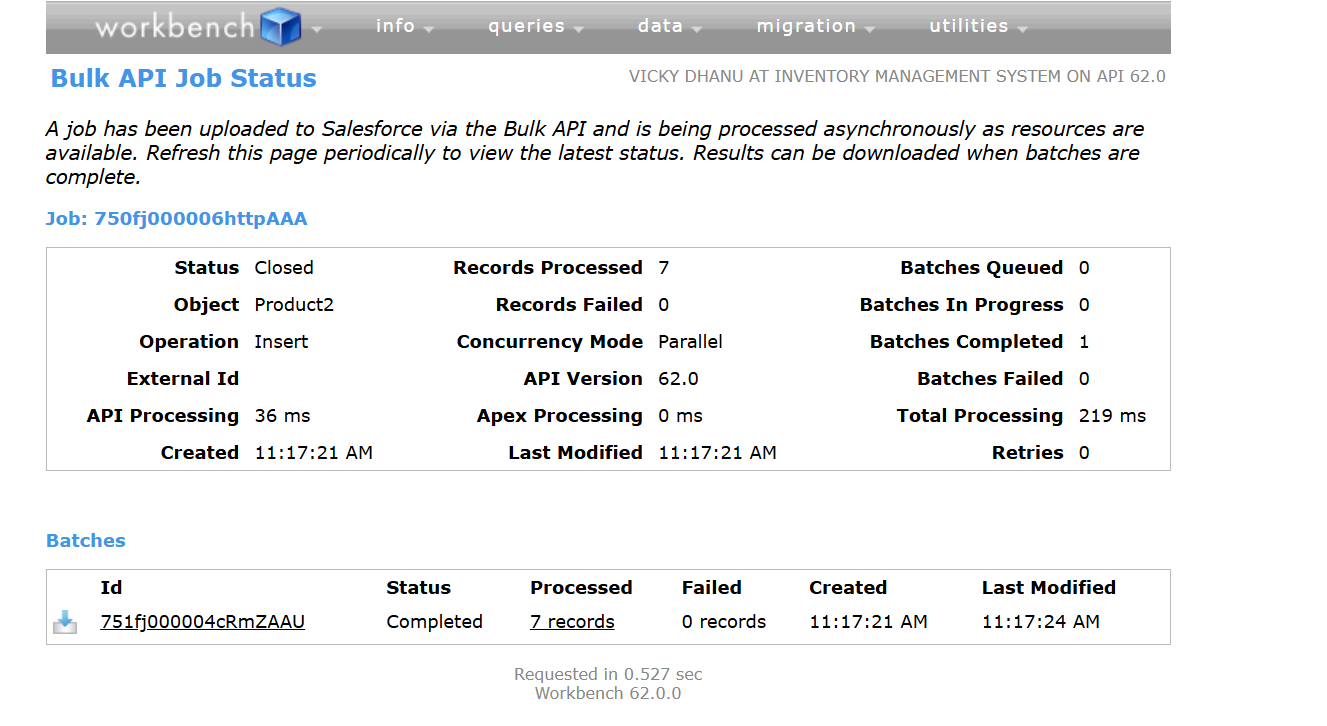
I have imported accounts using the data import wizard for the project  


Steps to use the data import wizard:

Go to setup search for Data Import.  
click on Data import wizard.  
select the objects to be imported.  
upload a csv or any file that is supported there.  
now check the mappings and edit if any mistakes.  
now finally click import the process happens and you can see whether the import is success or failure. **Workbench**

* Web-based tool for performing advanced Salesforce data operations.
* Supports CRUD operations: Insert, Update, Upsert, Delete, and Hard Delete.
* Allows working with both standard and custom objects.
* Can handle large datasets efficiently (more than 50,000 records).
* Supports SOQL queries and SOSL searches to retrieve Salesforce records.
* Provides bulk API support, allowing parallel processing of records.
* Enables field-level mapping and external ID usage for related objects.
* Offers detailed error reporting for failed operations.
* Useful for testing, troubleshooting, and bulk data migration.
* Requires a Salesforce login; works with both production and sandbox orgs.

I have imported the product data through the workbench which is a third party platform for data Import  




Phase 9: Reporting, Dashboards & Security Review  
**Reports**

* Tool for analyzing Salesforce data in a structured format.
* Supports standard and custom objects, showing records in tables, charts, or dashboards.
* Allows filtering, grouping, and summarizing data to extract insights.
* Types of reports include Tabular, Summary, Matrix, and Joined.
* Can be exported to Excel or CSV for offline use.
* Useful for tracking KPIs, monitoring performance, and supporting decision-making.
* Supports dynamic dashboards for visual representation of key metrics.

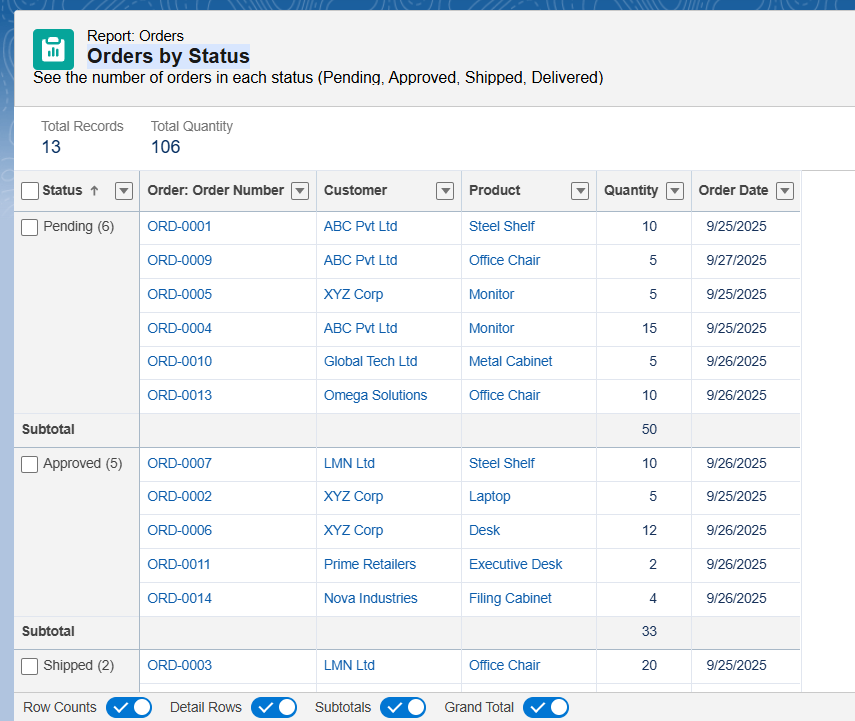
**Steps to create reports**  
 Go to the Reports tab in Salesforce

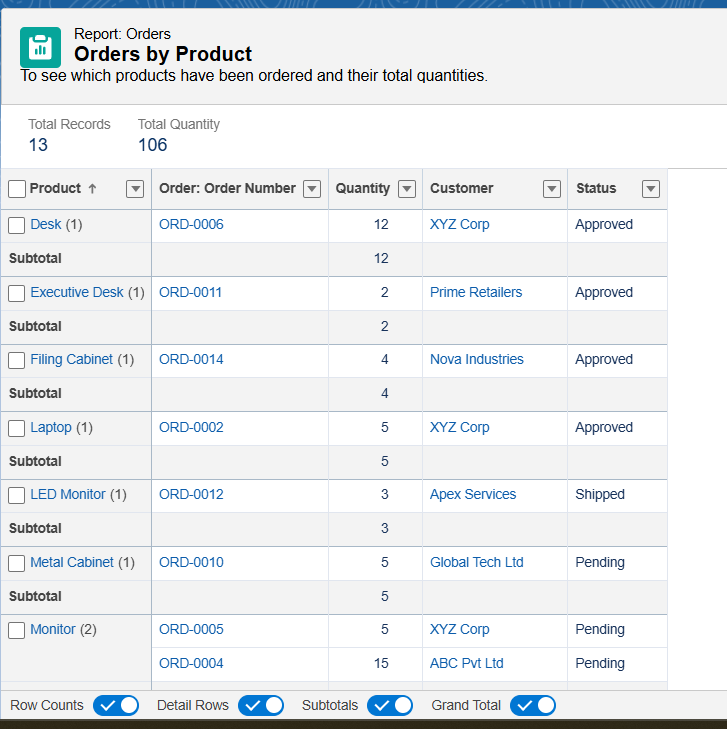
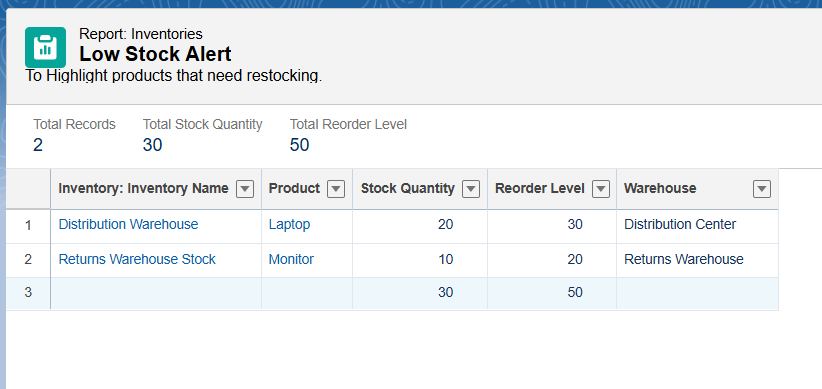
* Click New Report.
* Select the object (e.g., Orders, Inventory) and click Start Report.
* Add Filters to include only the relevant records.
* Group Data by fields if needed (e.g., by Product or Status).
* Add columns, summaries, or charts as required.
* Click Save & Run → give the report a name and folder.
* Optionally, export to Excel or CSV.

I have created the following reports in a folder and given read access to the Inventory Manager  
**Reports Created**

1. **Inventory Stock Levels** – To See current stock for each product in the warehouse
2. **Low Stock Alerts** – To Highlight products that need restocking.
3. **Orders by Product** – Displays all orders with details like product, quantity, account, and status; useful for tracking order fulfillment.
4. **Orders by status**– To see which products have been ordered and their total quantities.

Images of the reports  
Inventory Stock Levels  
****

Orders by Status  


Orders by Product  
  
  
Low stock alerts  


**Dashboards**

* Visual representation of Salesforce reports using charts, graphs, and tables.
* Helps track key metrics and KPIs at a glance.
* Can combine multiple reports in a single dashboard.
* Supports dynamic filters to view data by region, user, or time period.
* Enables real-time monitoring of business processes like inventory and orders.
* Useful for decision-making, performance tracking, and management reporting.

**Steps to Create a Salesforce Dashboard**

* Go to the Dashboards tab and click New Dashboard.
* Enter a Name and select a Folder to save it.
* Click Create, then click +Component to add a report.
* Select a report and choose a component type (e.g., chart, gauge, table).
* Configure the component settings (display, filters, grouping).
* Repeat to add multiple components as needed.
* Click Save and then Refresh to view real-time data.

**Dashboards Created  
Inventory & Orders Dashboard**Consolidates data from Inventory Stock levels, Low Stock Alerts, Orders by status, and Orders by Product.

Provides a single view of inventory levels and order status.

Highlights low-stock products for timely restocking.

Tracks all orders and identifies confirmed orders for operational monitoring.

Supports quick insights and decision-making for inventory management and order processing.

Visualizes data through charts, tables, and graphs for easy interpretation.  
  
The below is the image of my dashboard  
