Project Title: Restaurant CRM (Bookings + Orders + Loyalty)

Industry: Food & Beverage / Hospitality
Project Type: B2C Salesforce CRM Implementation
Target Users: Restaurant Managers, Staff (Booking & Order team), Customers, Marketing Team

Problem Statement

A restaurant chain receives numerous bookings and food orders daily via phone, website, and in-person visits. However:

- Bookings are tracked manually in registers or spreadsheets
- Duplicate or overlapping reservations often lead to poor customer experience
- Orders are not centralized, making it hard to link them with customers
- Loyalty programs are inconsistent and manually managed
- No real-time insights into top customers, peak booking hours, or revenue trends

■ To address these challenges, the restaurant wants to implement a **Salesforce CRM solution** to:

- Automate table bookings and send instant confirmation emails/SMS
- Manage customer orders and link them to accounts/contacts
- Track loyalty points and reward repeat customers
- Generate real-time reports for bookings, orders, and customer trends
- Improve customer communication through automated notifications

Use Cases

Booking Management

- Allow staff to create table bookings (date, time, party size, table number)
- Prevent double-booking using validation rules
- Send automatic confirmation/cancellation emails to customers

Order Management

- Record dine-in and take-away orders
- Link orders to customers (new or repeat)
- Track order details (menu items, price, quantity, bill amount)

Customer Management

- Maintain profiles with contact details, visit frequency, and order history
- Categorize customers (new, regular, VIP)
- Auto-assign loyalty program tiers based on spend

Loyalty Points Program

- Assign points per order (e.g., 1 point for every ■100 spent)
- Track accumulated points and redemption history
- Auto-notify customers of available rewards

Reporting & Dashboards

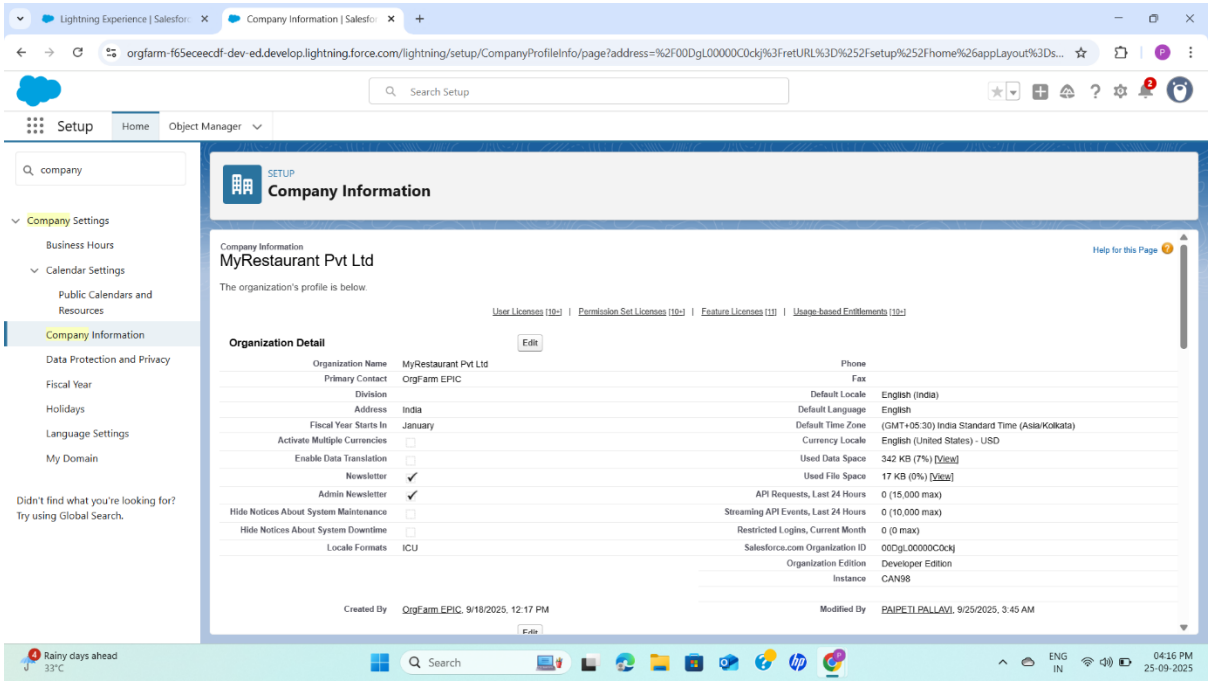
- Top 10 loyal customers by spend
- Peak booking hours and busiest days
- Revenue from dine-in vs. take-away
- Order trends per menu category

RESTAURANT CRM

Company Information

This is the foundation of any Salesforce organization. The *Company Information* section stores the official details of the business such as the company name, address, time zone, default currency, and fiscal year.

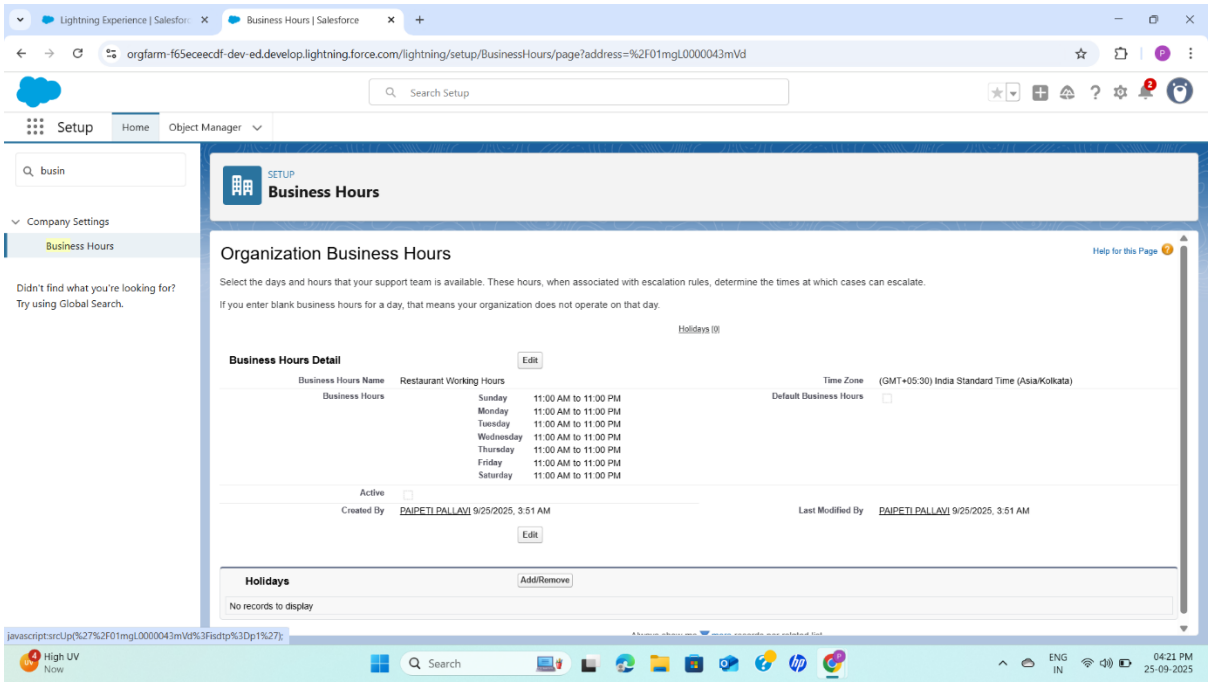
- **Organization Name**: Defines how the system recognizes your company and how it will appear in system headers, licenses, and reports. For example, *Restaurant CRM Pvt Ltd*.
- **Default Time Zone**: Ensures that all date/time stamps, workflows, and automation rules work according to the company's working hours. For India, we use (GMT+05:30) India Standard Time.
- **Default Currency**: Determines the default currency for opportunities, reports, and transactions. Businesses that operate in multiple regions can enable **multi-currency**.
- **Fiscal Year**: Critical for reporting revenue and performance. Salesforce supports both **Standard Fiscal Years** (Jan–Dec) and **Custom Fiscal Years** (e.g., Apr–Mar for Indian businesses).



2. Business Hours

Defining *Business Hours* in Salesforce is vital because many service-related features, like **case escalations, SLA timers, and workflow triggers**, depend on them.

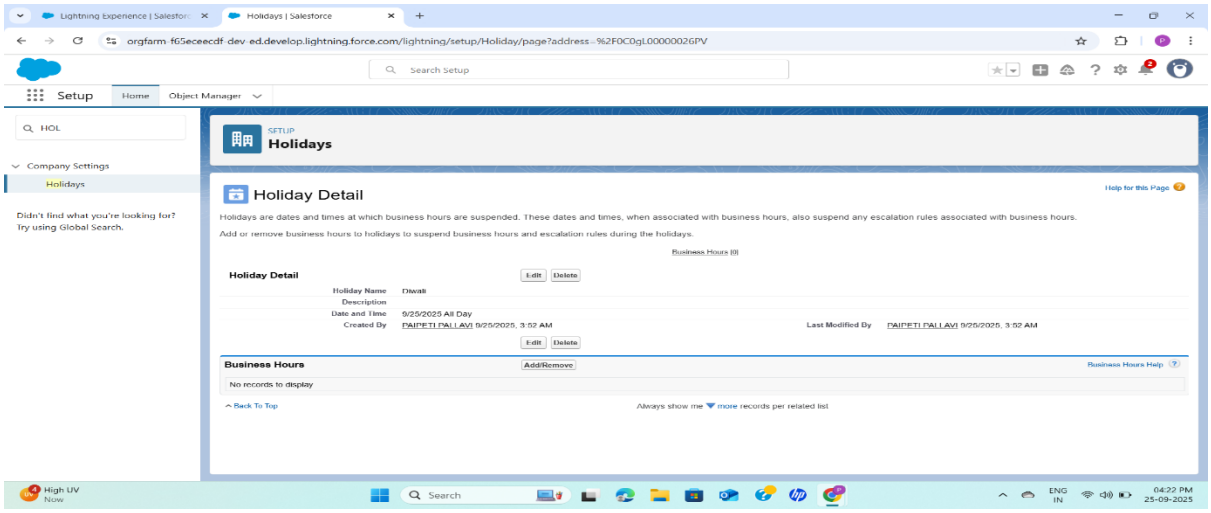
- Example: The restaurant operates from **11:00 AM to 11:00 PM** every day.
- Marking them as **default** ensures they apply across customer support cases and automation flows unless overridden.
- Having clear business hours helps track SLA commitments more accurately.



3. Holidays

Holidays allow you to define non-working days in Salesforce. These are linked to business hours to ensure SLA calculations and workflows don't count those days.

- **Example: *Diwali – Nov 12, 2025*.**
- **When a holiday is marked, Salesforce automatically pauses SLA timers and escalations on that day.**
- **This ensures accurate reporting and avoids penalizing the support team during holidays.**

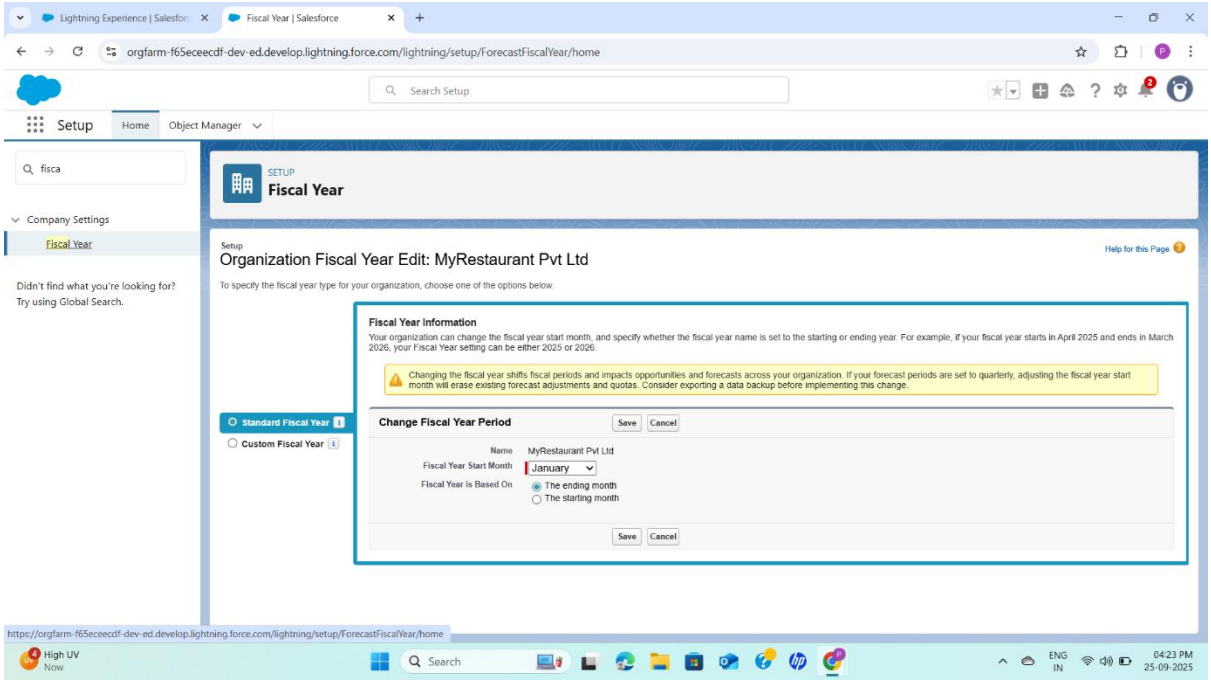


4. Fiscal Year

The *Fiscal Year* setup directly affects **reporting, forecasting, and dashboards**.

- **Standard Fiscal Year**: Runs Jan–Dec. Recommended for global organizations unless specified otherwise.
- **Custom Fiscal Year**: Can follow any 12-month pattern (e.g., Apr–Mar for Indian businesses). This is useful for aligning Salesforce reporting with statutory accounting requirements.

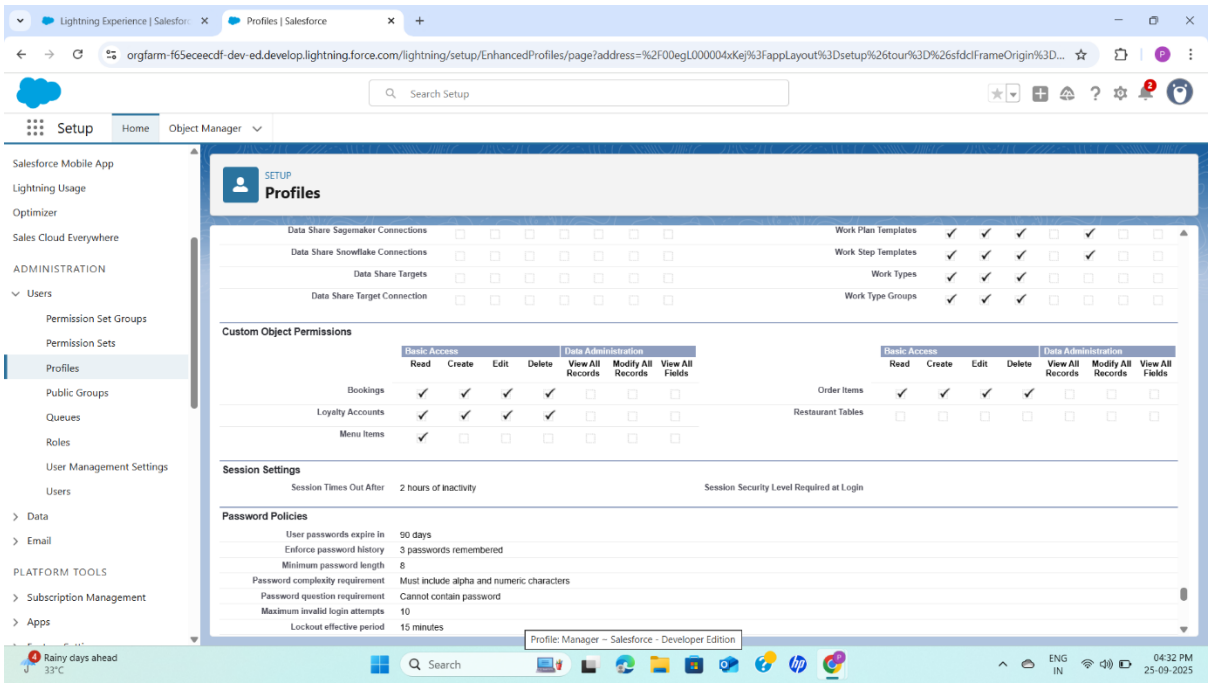
Once a fiscal year type is enabled, it affects all company reports, so the decision must be made carefully.

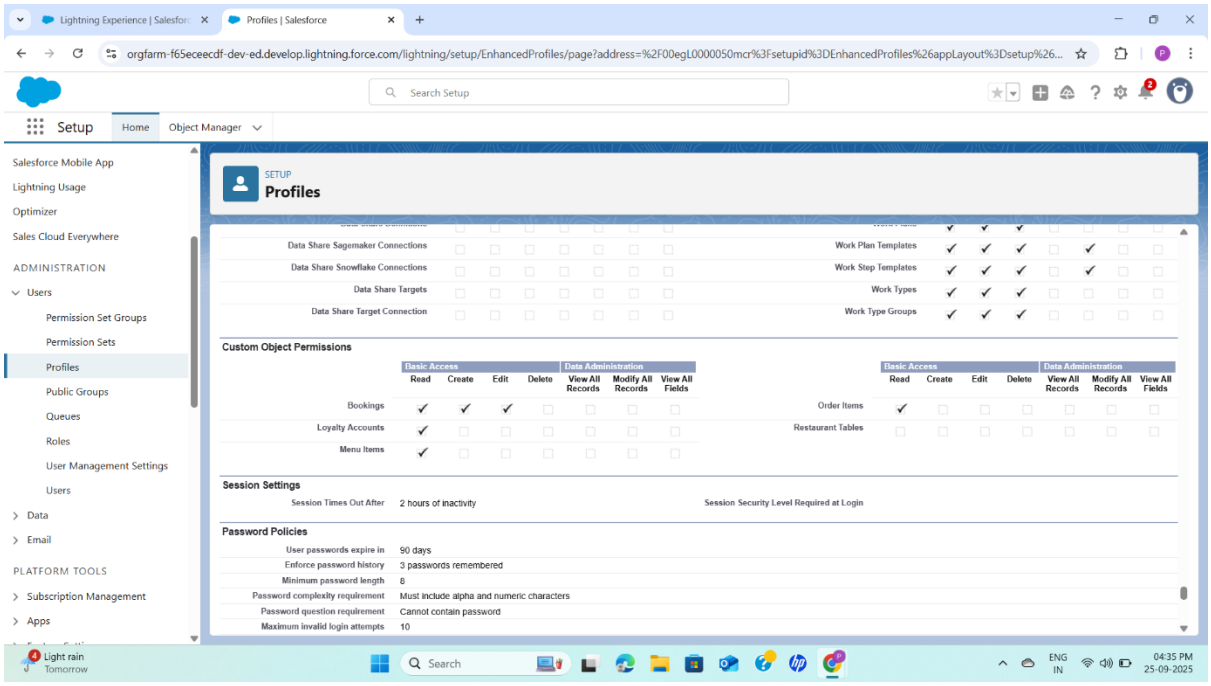


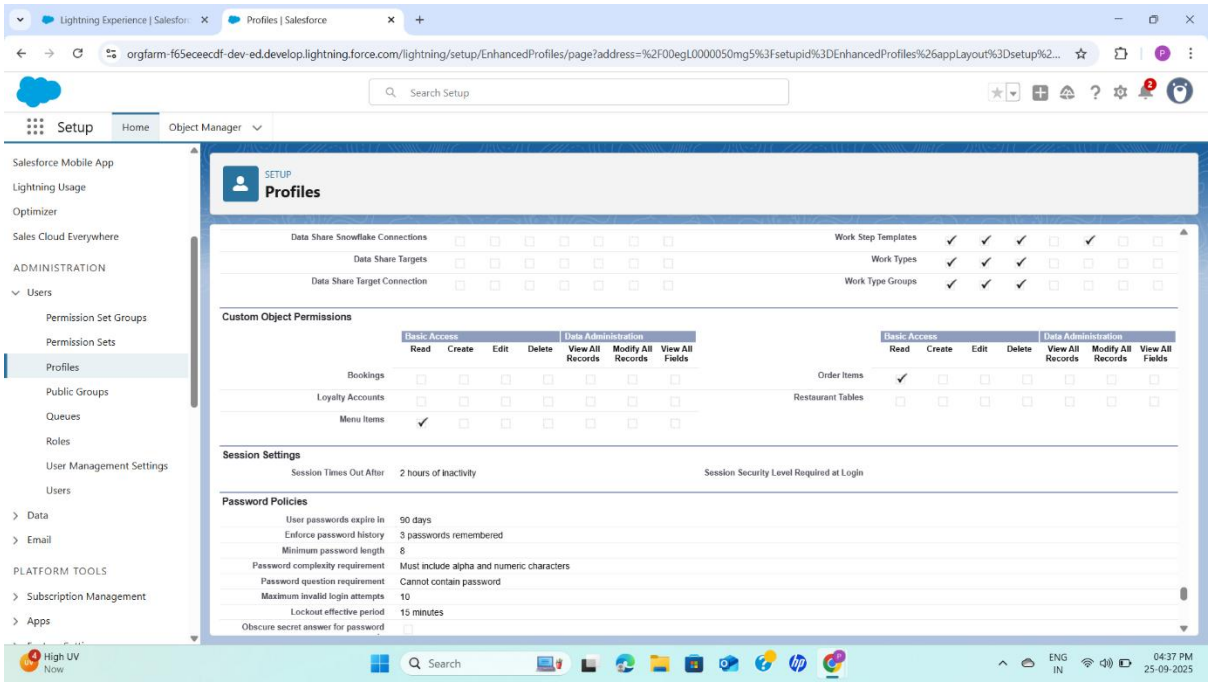
5. Profiles

Profiles are the backbone of Salesforce's **security model**. They define **object permissions, field-level access, page layouts, and app visibility**.

- **Manager Profile** → Full CRUD (Create, Read, Update, Delete) on Orders, Bookings, and Loyalty objects.
- **Host Profile** → Can create and manage bookings/orders but has limited edit access.
- **Kitchen Profile** → Read-only access to orders and related order items.



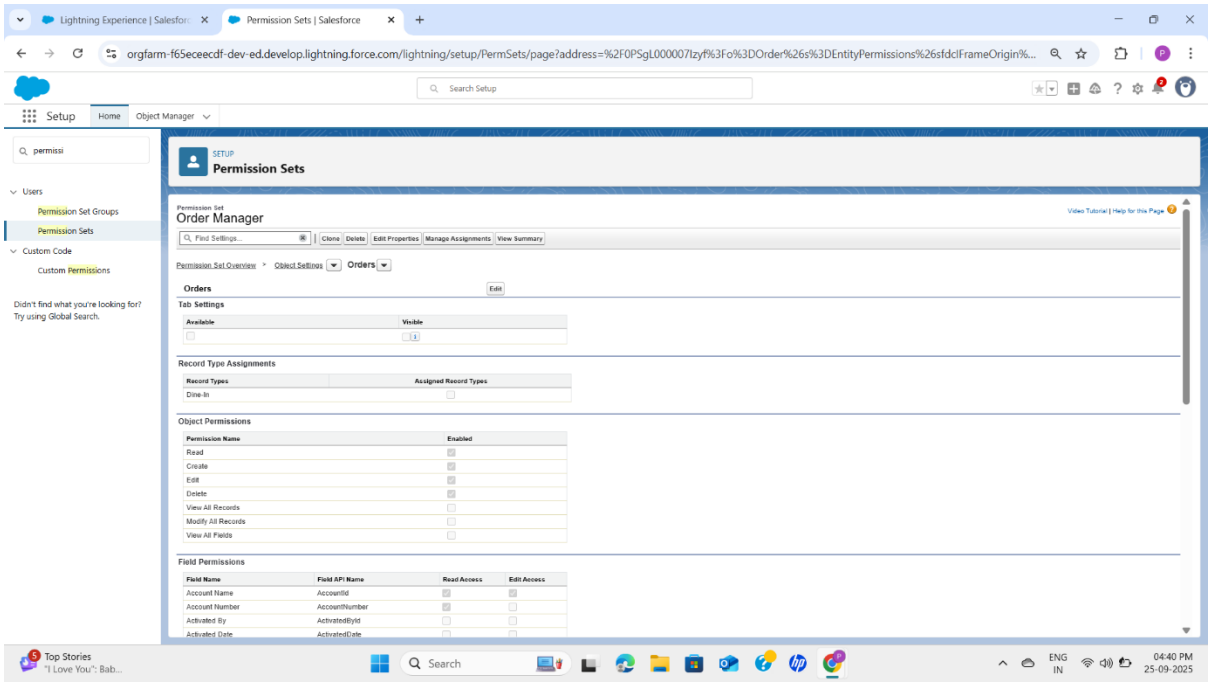




6. Permission Sets

Permission Sets allow granting **additional access on top of profiles** without changing the base profile.

- Example: *Order_Manager* permission set → Grants full access (Read, Create, Edit, Delete) on the Order__c object.
- Assignable to any user when temporary or extended access is required.
- This provides flexibility in managing security without cloning multiple profiles.

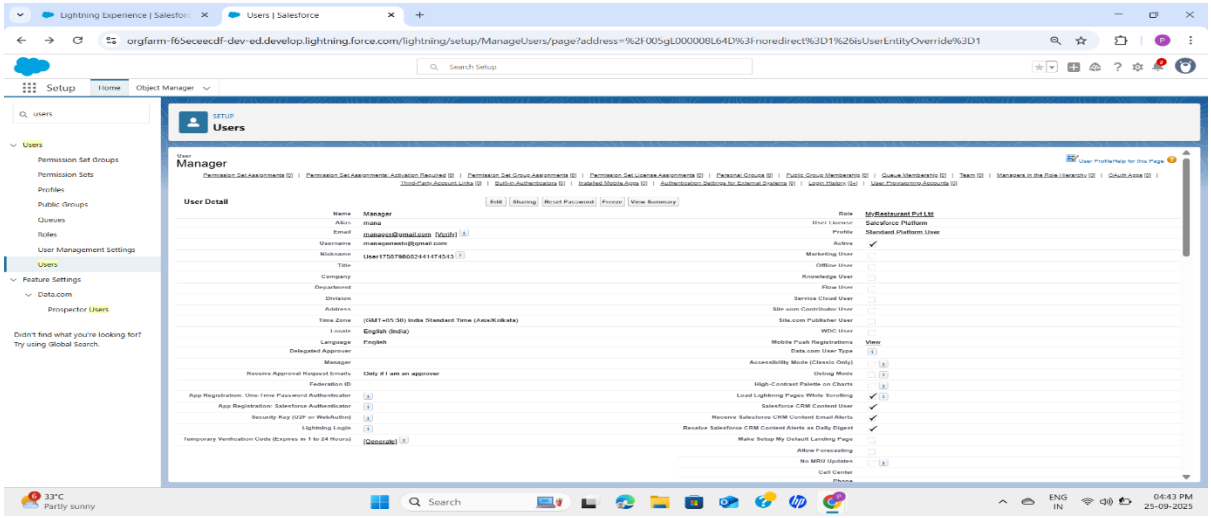


7. Users

Users represent real people who log in to Salesforce. Each user must be assigned a **profile** and optionally **permission sets**.

- Example: *Manager User* with Manager Profile.
- Each user consumes a license, which defines the type of functionality they can access (e.g., Sales Cloud, Service Cloud, or Platform license).

Proper user setup ensures accountability and controlled access to the system.

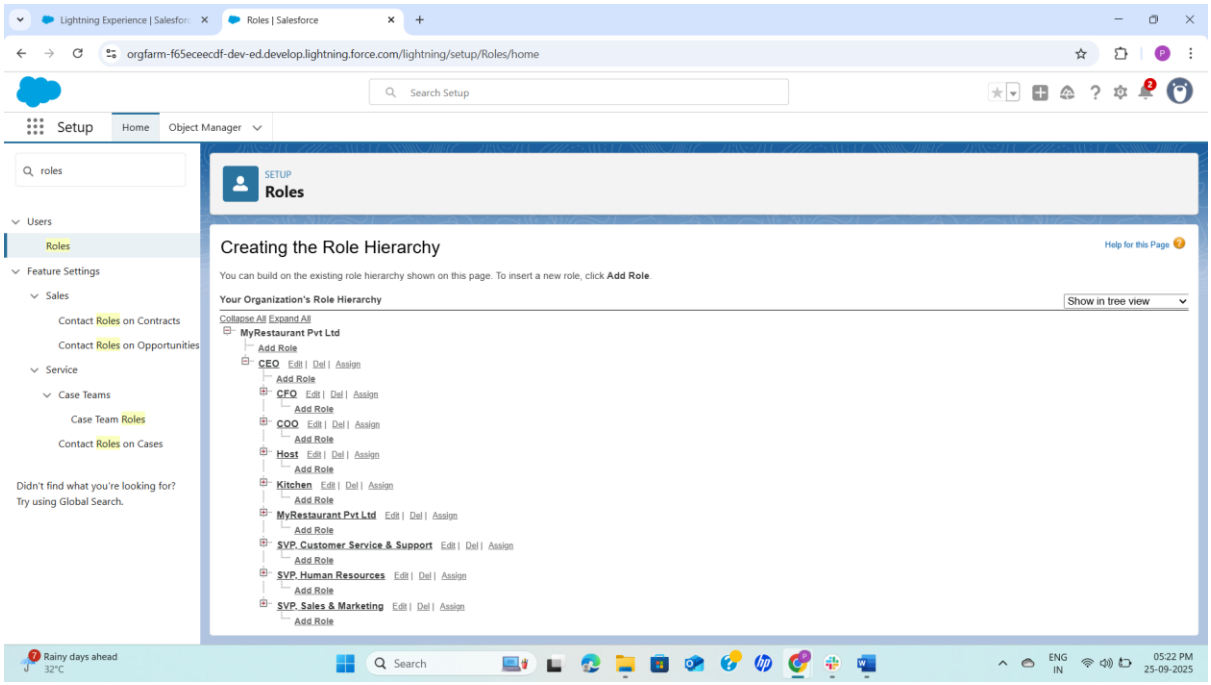


8. Role Hierarchy

Roles control **record visibility** and define how data rolls up in the organization.

- Top role: **Restaurant Manager** → Can view and manage all subordinate data.
- Under: **Host** → Handles bookings.
- Under: **Kitchen** → Sees order details only.

The role hierarchy supports Salesforce's **"record sharing up the hierarchy"** rule, meaning managers can automatically see the records owned by their subordinates.

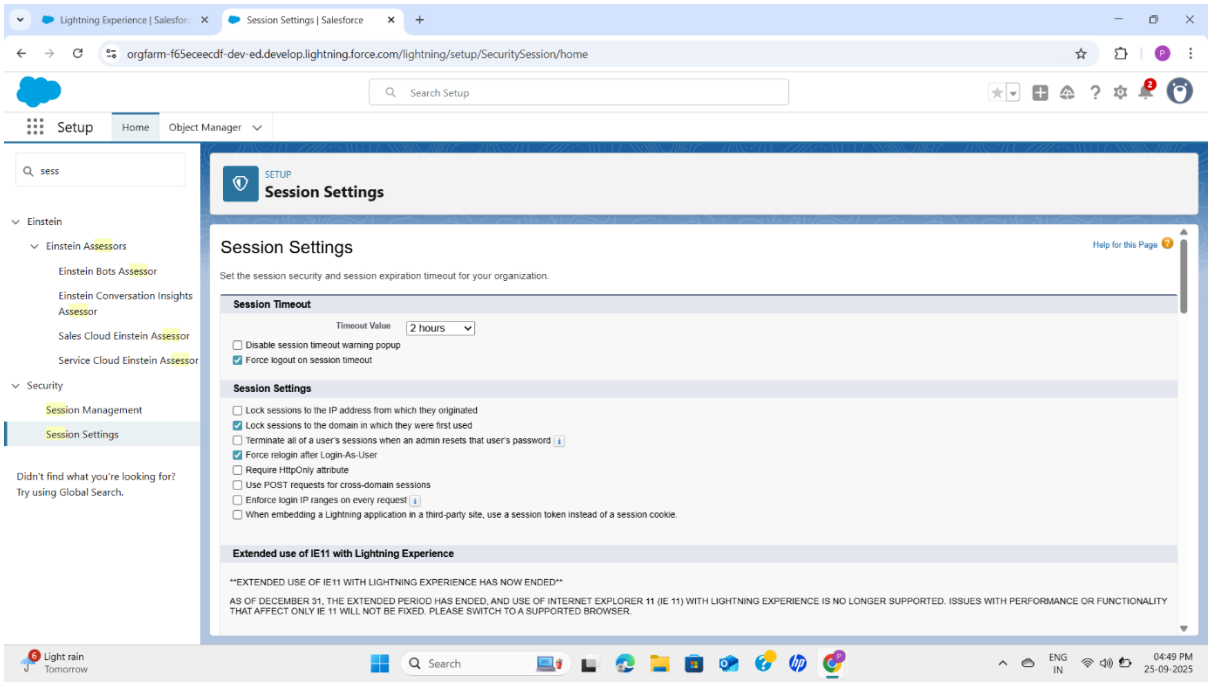


9. Session Settings

Session settings define the **security rules** for how long users remain logged in and how sessions are handled.

- **Force re-login after logout**: Prevents unauthorized re-entry.
- **Timeout (2 hours)**: Ensures inactive sessions automatically log out for security.
- **Lock sessions to IP address**: Optional, prevents session hijacking but may inconvenience mobile/remote workers.

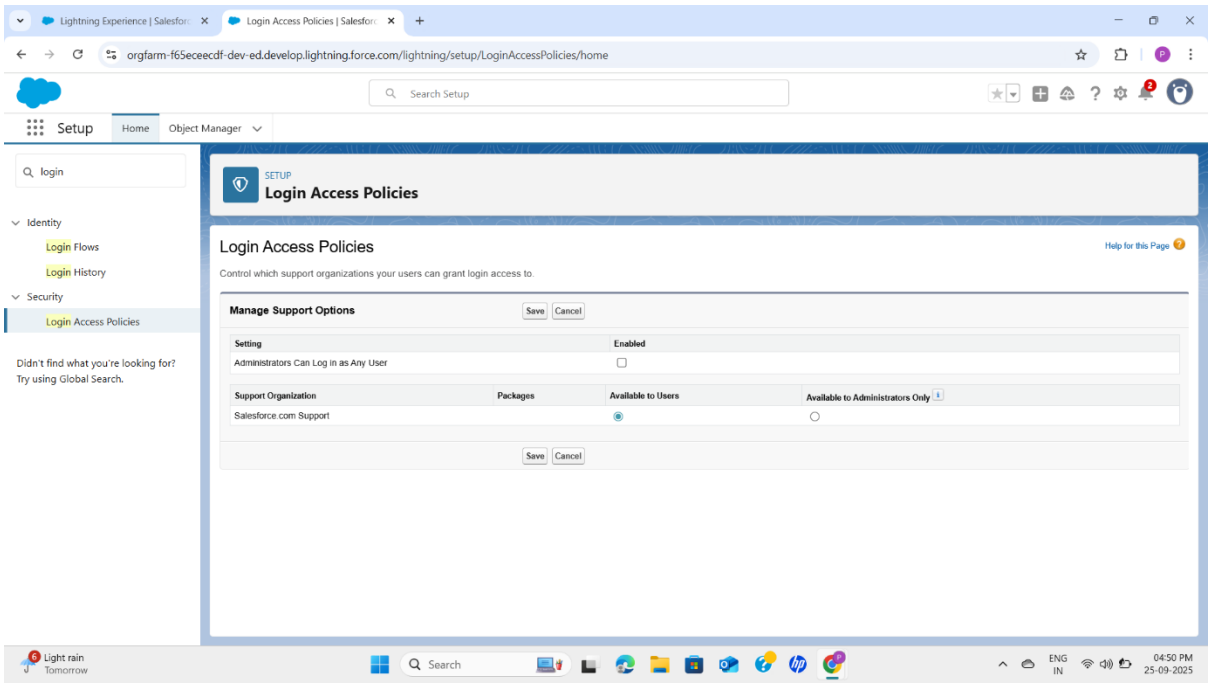
This setup balances **security with usability**.



10. Login Access Policies

This feature allows admins to log in as other users (with permission) for **troubleshooting, testing, and training**.

- Example: The admin can log in as a Kitchen User to verify if permissions and page layouts are working correctly.
- This is critical during testing before going live.



Phase 3 — Data Modelling & Relationships

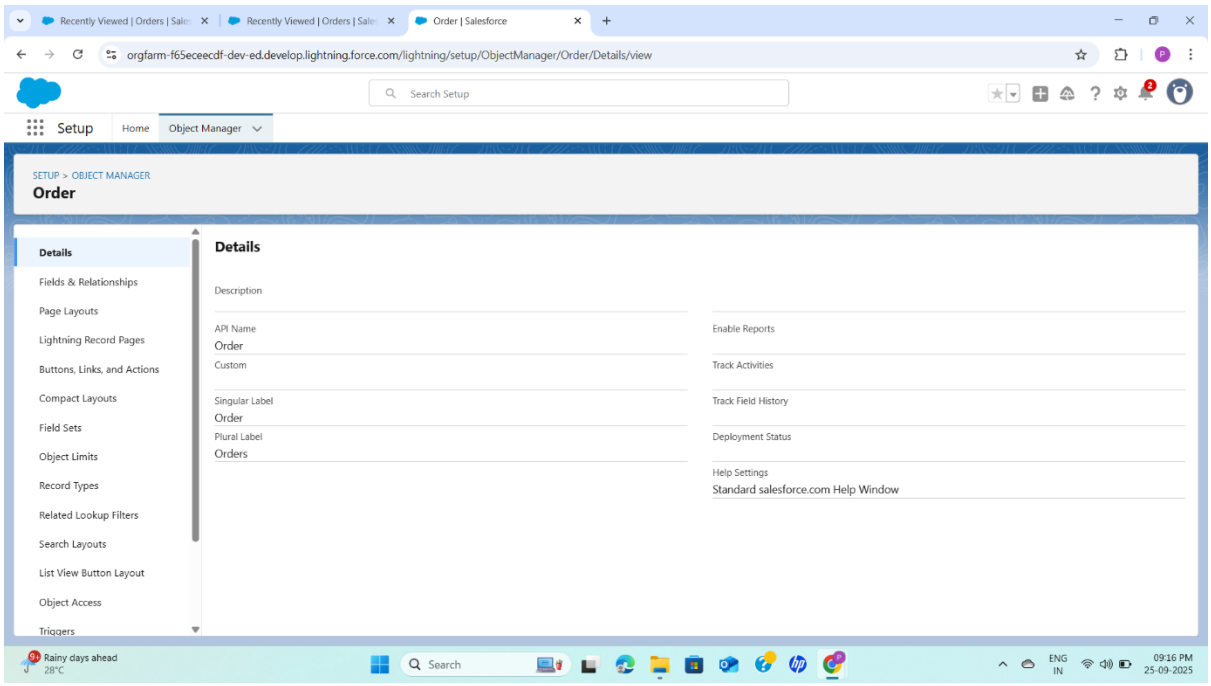
In Salesforce, data modelling is the process of defining **objects, fields, and relationships** so that business processes can be represented properly in the system. For a restaurant CRM, the core entities include Orders, Order Items, Bookings, and Loyalty Accounts.

Custom Objects

Custom objects are created to represent real-world entities. In this case:

- **Order** → Captures details of a customer's order.

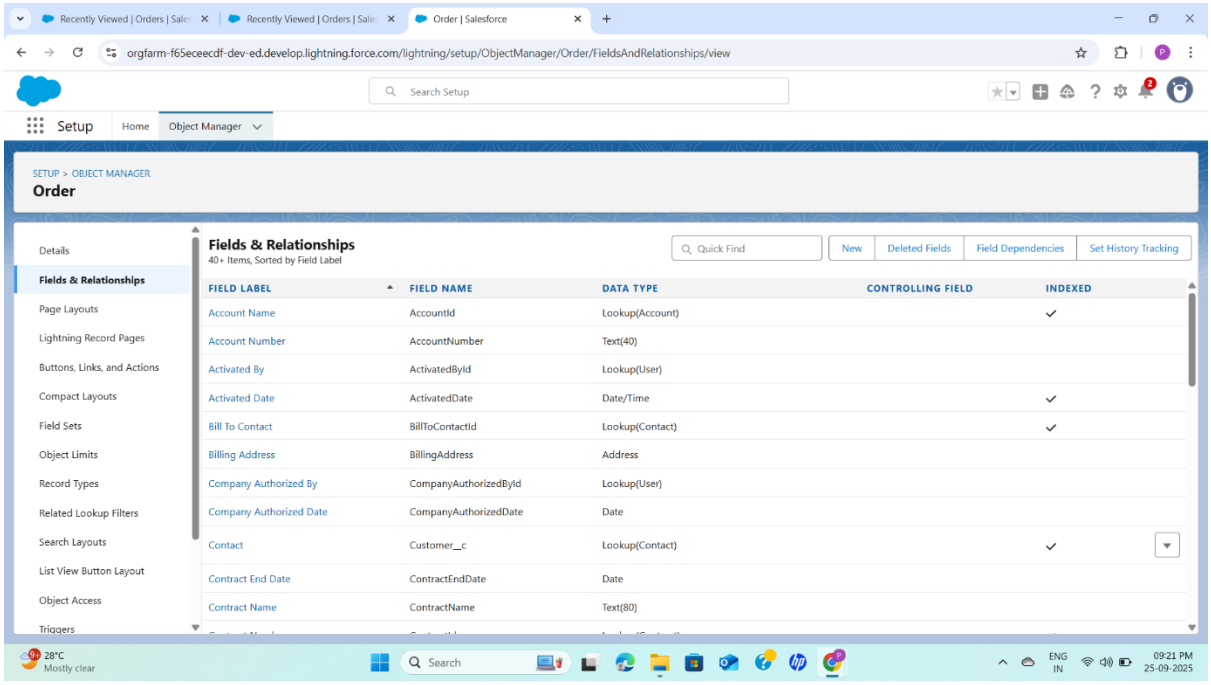
These objects are configured with reporting, activities, and field history tracking enabled so data can be monitored and analysed.

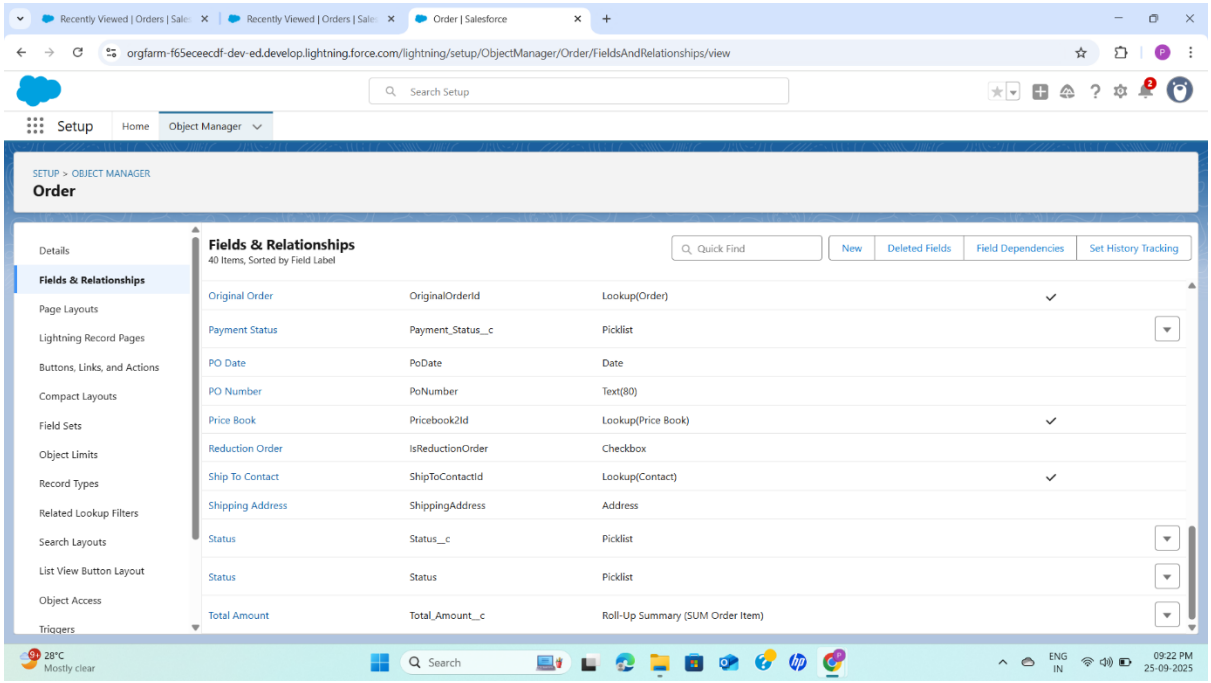


The **Order object** captures all information about a customer's food order. It acts as the central business record for sales transactions in the restaurant. Each order may be tied to a **customer (Contact)**, a **Booking** (if it was pre-reserved), and contains multiple **Order Items** (individual dishes).

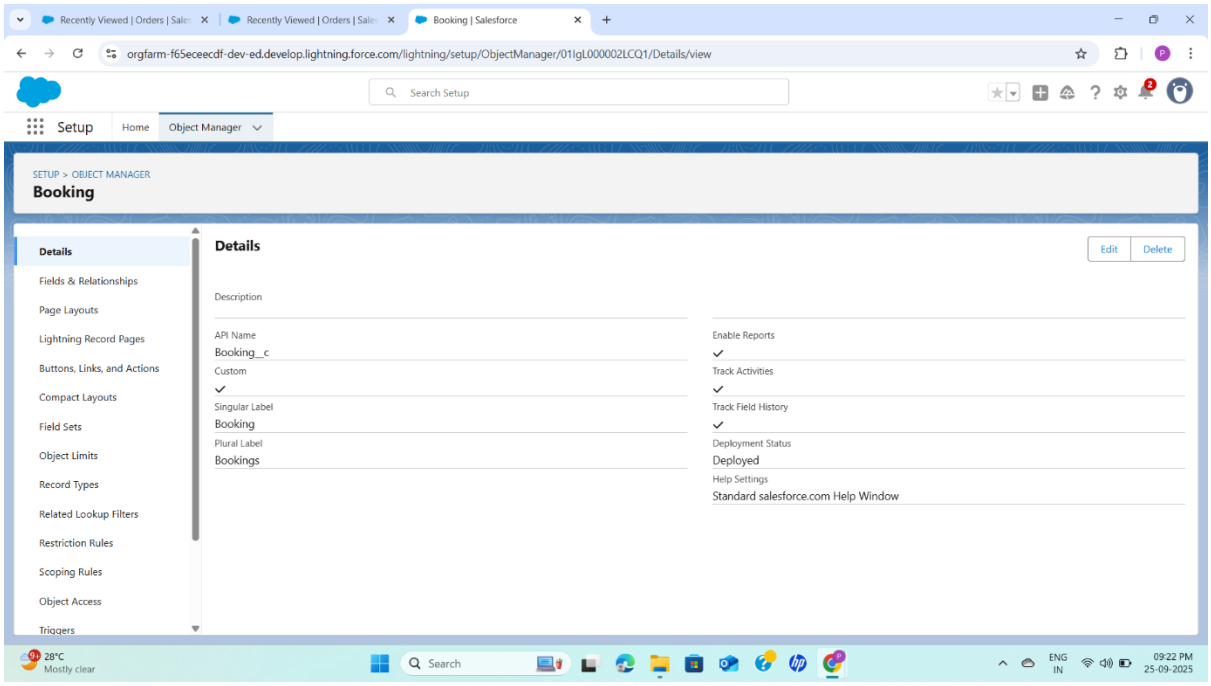
- The **Order Type** field differentiates whether the order is Dine-In, Takeaway, or Delivery.
- The **Status** field shows the order lifecycle (e.g., Draft → Placed → Preparing → Served → Paid → Cancelled).
- **Payment Status** ensures financial tracking, while **Total Amount** provides billing details.
- If delivery is chosen, an address is also stored.

Key Relationship:
Orders are **parents** to Order Items in a Master-Detail relationship, ensuring that if the order is deleted, all its line items are removed. This also allows roll-up summaries of order totals.





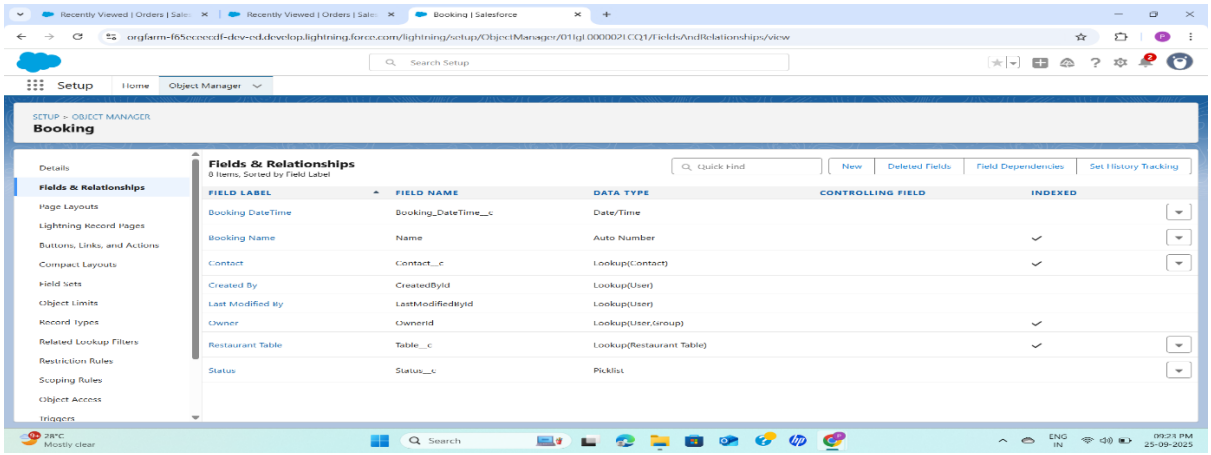
Booking → Stores reservations made by customers.



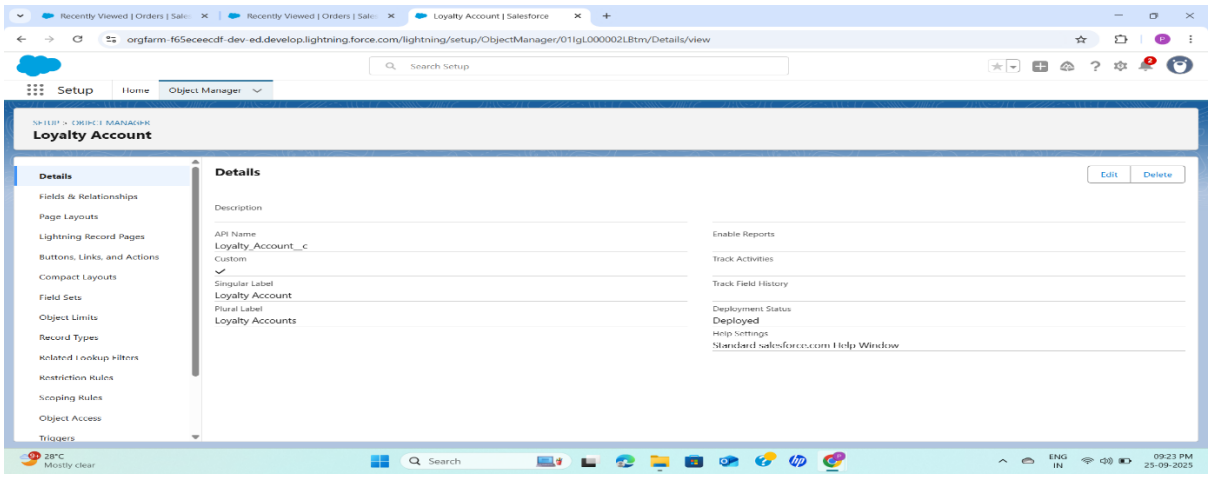
The **Booking object** manages restaurant reservations. It allows customers to reserve a table for a specific time slot and number of guests.

- The **Booking Time** defines when the reservation occurs.
- **Pax** indicates the number of people.
- **Status** reflects the booking lifecycle (Confirmed, Seated, NoShow, Cancelled).
- Each booking is linked to a **Contact** (customer) and to a **Restaurant Table** to manage seating capacity.

Key Role:
Bookings help prevent overbooking, improve customer service, and give staff visibility into restaurant occupancy.



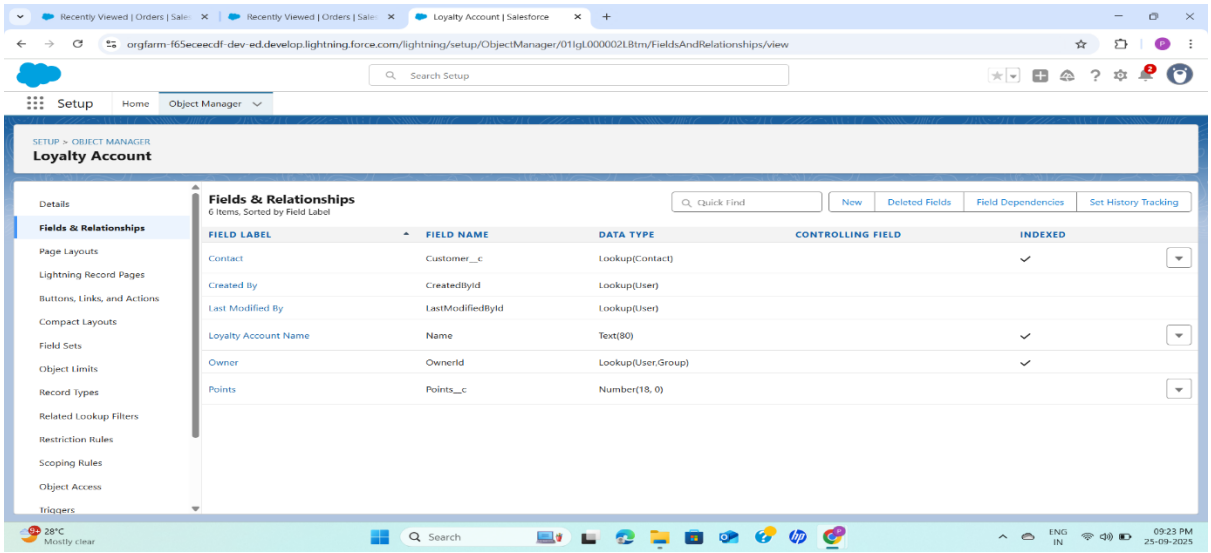
Loyalty Account → Tracks points and tier for customer loyalty.



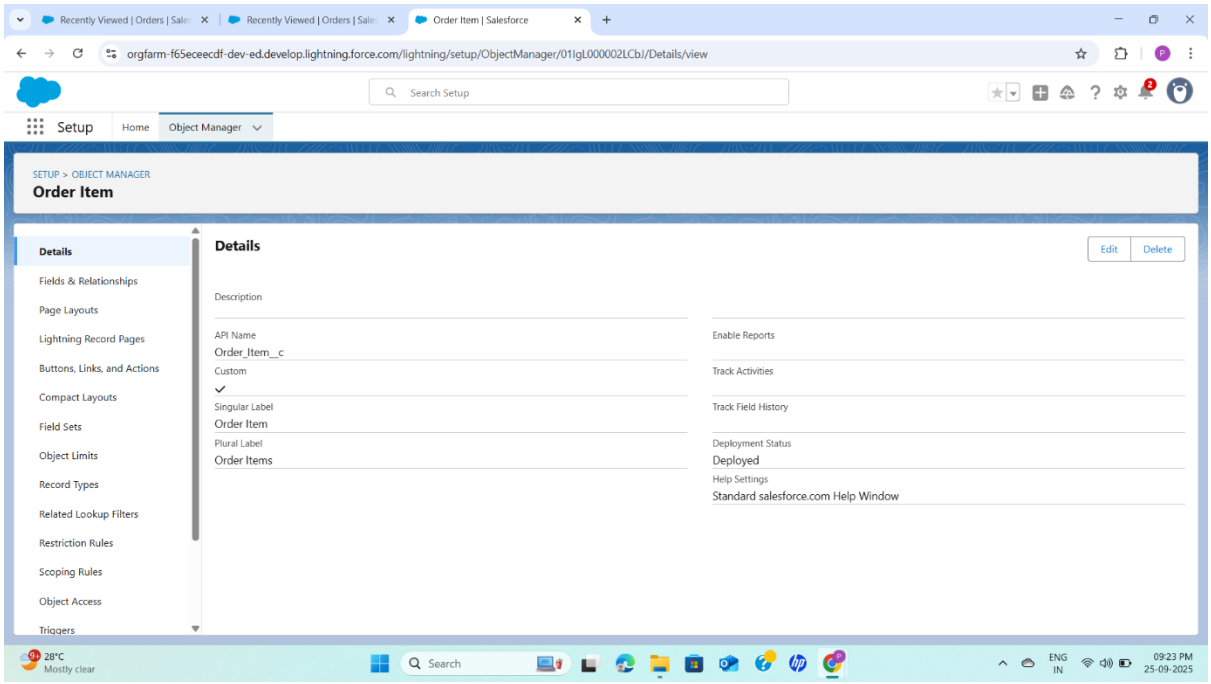
The **Loyalty Account object** tracks customer engagement through points and reward tiers. It is tightly linked to the **Contact** object (the customer).

- Each customer can have a **Loyalty Account** where points are credited based on orders.
- **Points Balance** keeps track of available reward points.
- **Tier** (Bronze, Silver, Gold, etc.) reflects customer loyalty level.
- **Join Date** stores when the customer enrolled in the loyalty program.

Key Role:
This object supports marketing and customer retention strategies by incentivizing repeat business.



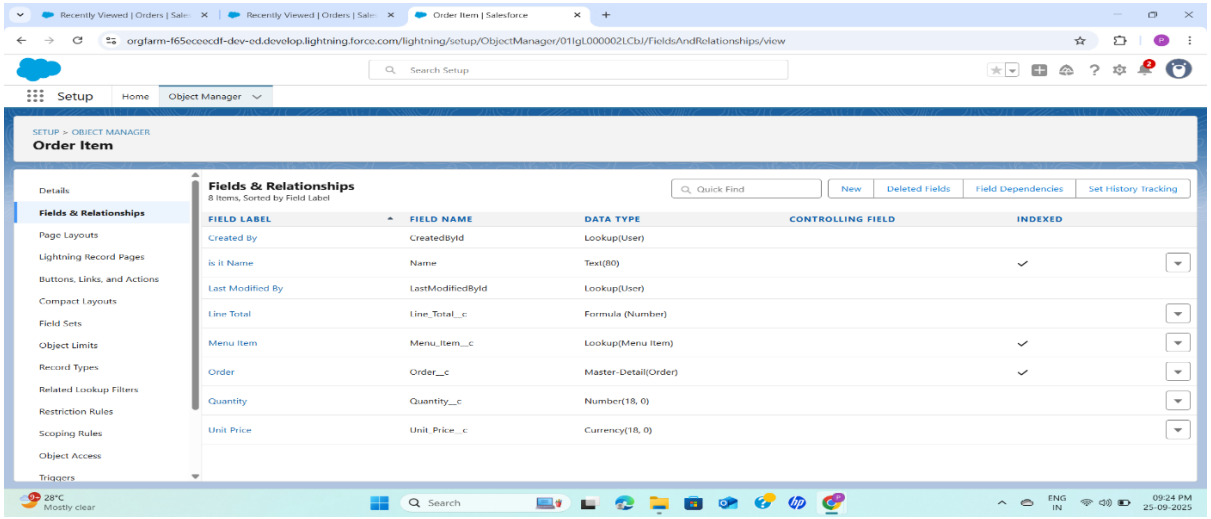
Order Item → Child of Order, representing individual menu items within the order.



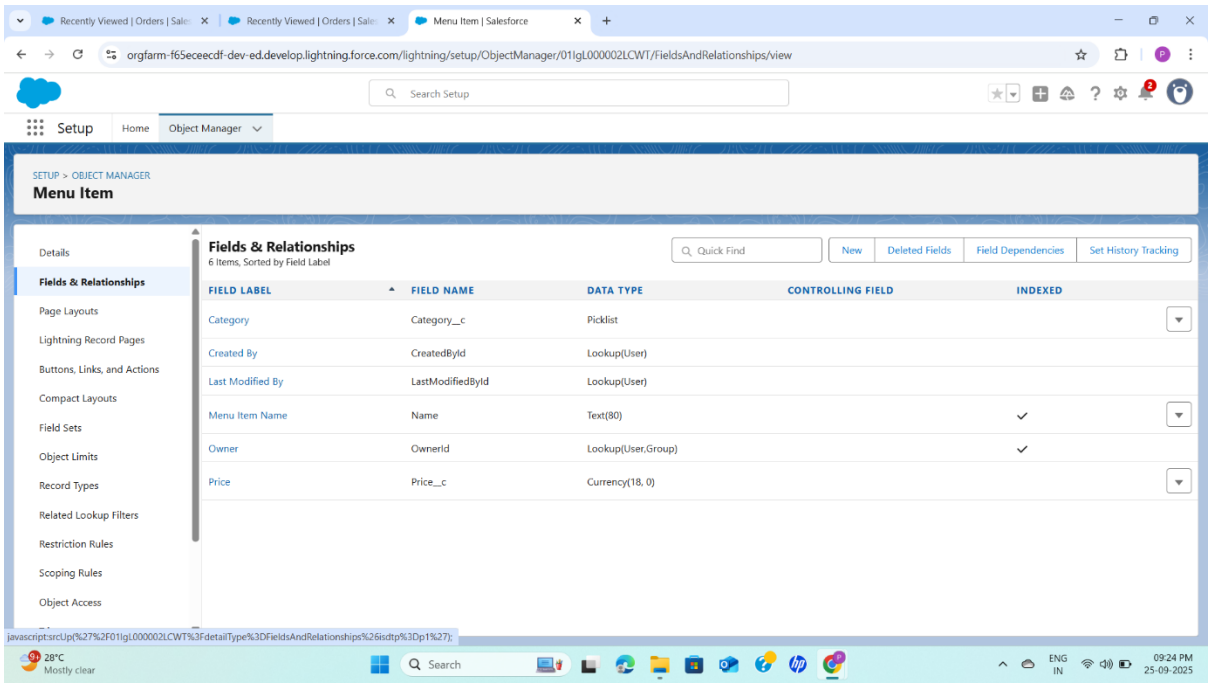
The **Order Item object** represents the specific menu items within an order. It ensures detailed tracking of what was ordered, in what quantity, and at what price.

- It is **child** to Order (via Master-Detail).
- It links to the **Menu Item object**, which contains the catalog of dishes available.
- **Quantity** and **Unit Price** define the financial details, while **Line Total** (Quantity × Price) ensures accurate calculation per item.

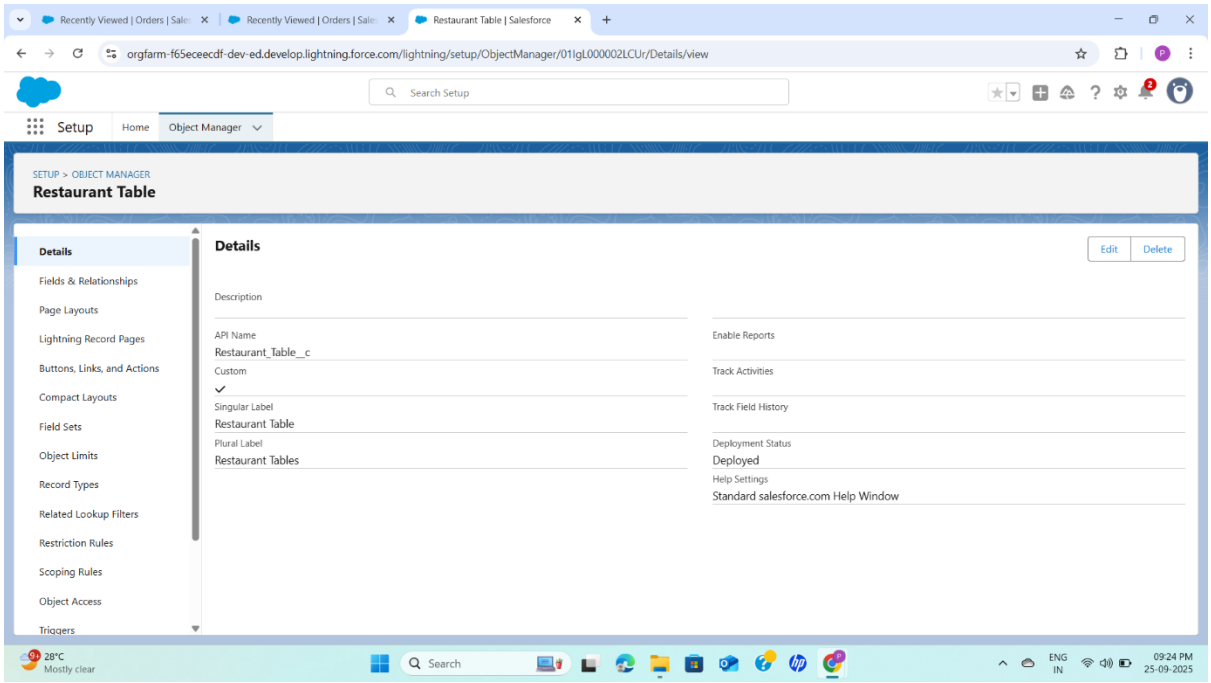
Key Role:
Without Order Items, the Order object would only represent a lump sum; Order Items break it down into granular details for billing, kitchen preparation, and inventory.



There are some fields and relationships in the menu item that are given below



The **Restaurant Table** object represents the physical seating arrangement in the restaurant. It helps the system track which tables are available, reserved, or occupied during a booking. This ensures efficient seat management and prevents double-booking.

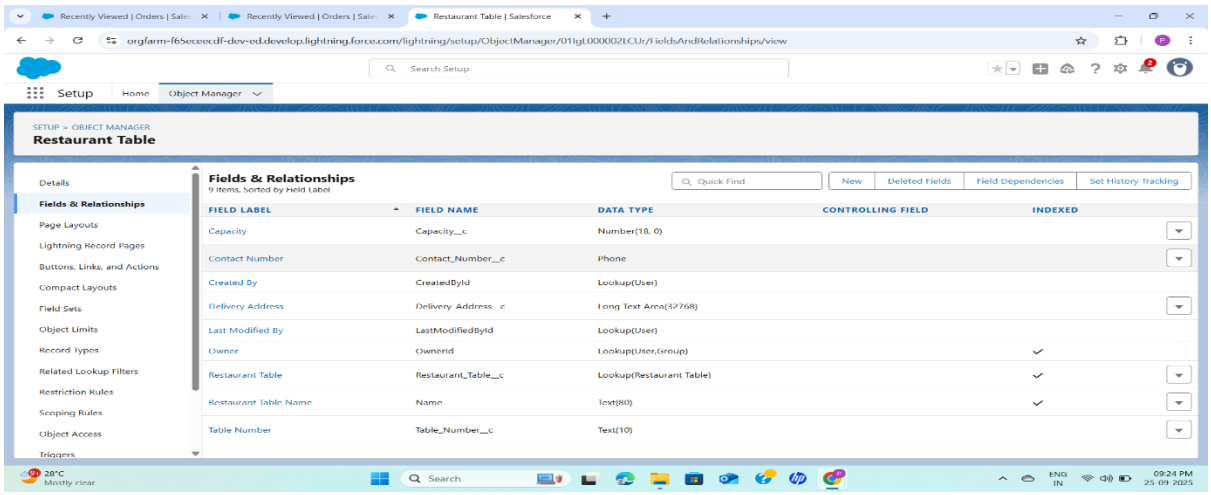


The **Restaurant Table object** represents the physical tables in the restaurant. It ensures that seating arrangements are managed efficiently.

- Each table has a **unique identifier (Table Number)**.

- **Capacity** defines how many guests it can accommodate.
- **Status** indicates whether a table is Available, Reserved, Occupied, or Out of Service.
- **Location** can differentiate indoor vs outdoor seating.

Key Relationship:
Bookings are linked to Restaurant Tables, ensuring that reservations are tied to real seating capacity. This avoids double-booking and helps staff assign tables efficiently



Create Page Layouts

Go to **Object Manager → Order__c → Page Layouts**.

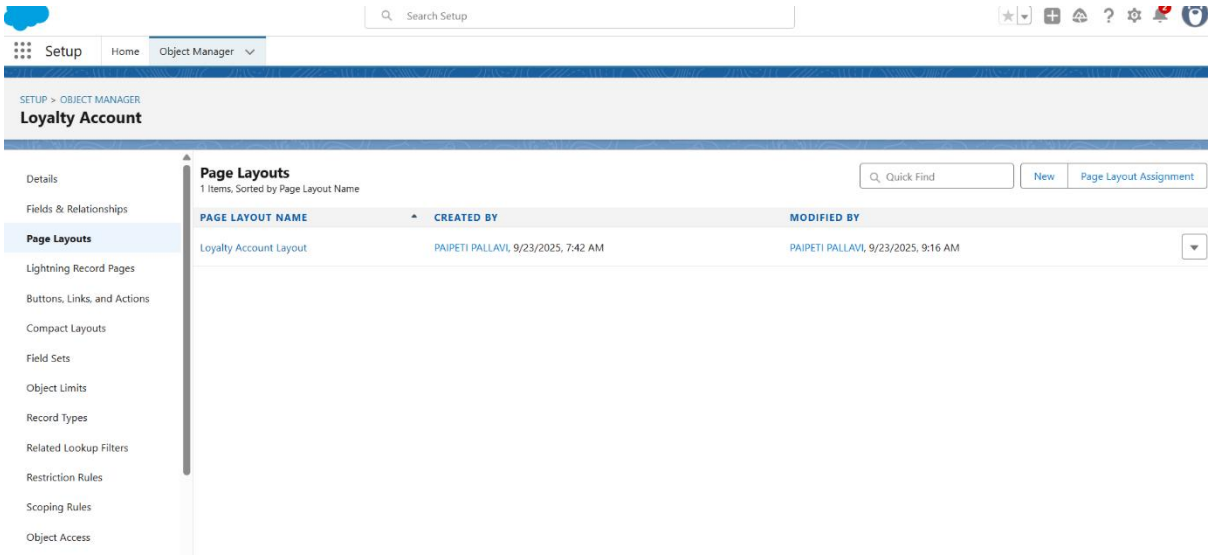
Create/Clone 2 layouts:

Order Layout – Dine-In

- Key fields: Table, Booking, Order Items, Status.

Order Layout – Delivery

- Same as above + **Delivery Address, Contact Phone, Delivery Notes**.



Creating Tabs for Custom Objects in Salesforce

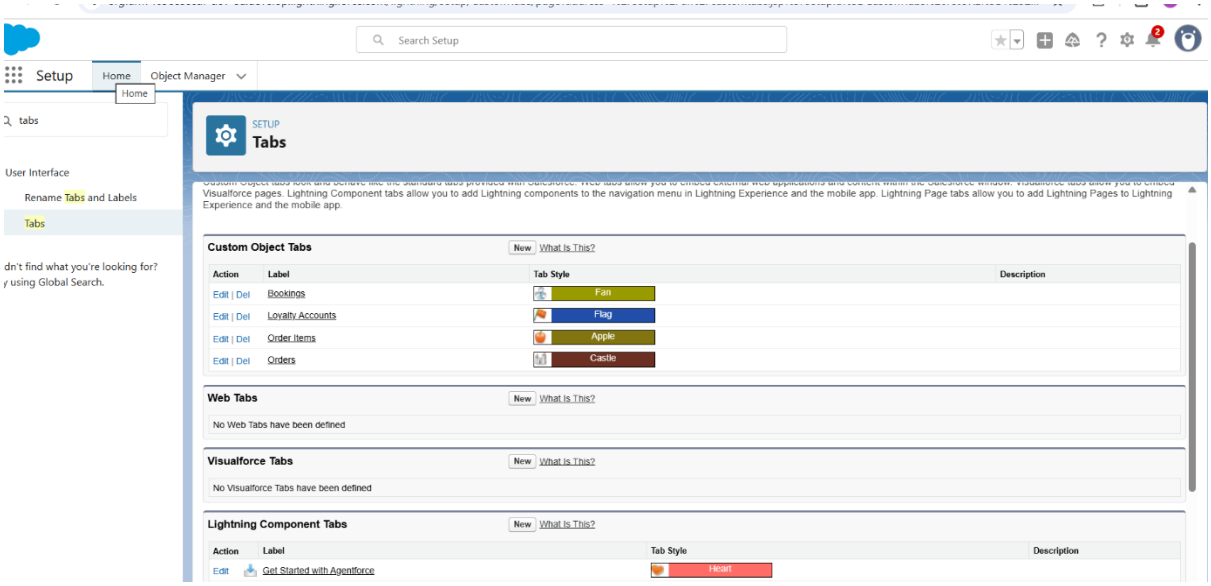
A **tab** in Salesforce is a user interface element that provides direct access to objects, records, or web resources from the Salesforce app's navigation bar. Think of a tab as a "doorway" into the database — without it, a custom object may exist but users cannot reach it easily from the UI.

The Booking tab makes reservations discoverable to hosts and managers. Picking an obvious Tab Style (calendar icon) is a small but important UX affordance — it reduces cognitive load so staff can scan navigation quickly. When you set Tab Visibility to "Default On" for Manager and Host but "Hidden" for Kitchen, you're enforcing a simple principle: the host and manager need immediate access to create and change bookings, the kitchen does not.

Orders are the central operational object for the restaurant, so their tab should be universally available to everyone involved in order handling (Manager, Host, Kitchen). Setting Tab Visibility to Default On for Manager, Host and Kitchen makes sense because hosts create orders, kitchen staff need to pick up items from the order, and managers monitor/override. The Orders tab provides access to list views (e.g., "Open Orders", "Pending Payment") and record pages (where order details, items and status live).

Order items are often a child object shown on the Order page as a related list, so creating a standalone Order Items tab is a deliberate admin decision: it's useful for data management, reporting ad-hoc edits, and bulk imports/cleanups but often unnecessary for everyday order-taking workflows. When you make the tab Default On for Manager and Optional or Read-Only for Host/Kitchen, you are signalling that managers are the ones who might need direct access to the raw line-item data (for auditing, refunds, adjustments) while service staff generally work from the Order parent record and its related list.

Loyalty and rewards often contain personal and behavioural data and so deserve conservative visibility. By making the Loyalty Accounts tab Default On for Manager, Read-Only for Host and Hidden for Kitchen you protect customers' reward data while still enabling hosts to view balances when assisting customers. This map supports common workflows: marketing and managers analyse and update tiers, hosts can check balances for in-person redemptions, and kitchen staff have no operational need for that information.



Schema Builder

Contact

- **Core Role:** Customer record.
- **Relationships:**
  - Lookup → **Loyalty Account** (one-to-one).
  - Lookup → **Order** (one-to-many).
  - Lookup → **Booking** (one-to-many).
- **Theory:**
  The Contact object is the **foundation of CRM**. Every customer interaction begins here — whether booking a table, placing an order, or redeeming loyalty points.

Loyalty Account

- **Core Role:** Tracks rewards/points for each customer.
- **Relationship:** Lookup → **Contact**.
- **Theory:**
  Enables the restaurant to manage customer retention by offering a loyalty program. Every order linked to a loyalty account can earn or redeem points.

Booking

- **Core Role:** Stores table reservations.
- **Relationships:**
  - Lookup → **Contact** (who made the booking).
  - Lookup → **Restaurant Table** (where the booking is seated).
- **Theory:**
  Ensures efficient **table allocation and reservation management**, preventing double bookings and improving service flow.

Restaurant Table

- **Core Role:** Represents physical seating arrangements.
- **Relationship:** Lookup → **Booking**.
- **Theory:**
  Allows staff to map reservations to available tables, track occupancy, and manage seating efficiently.

Order

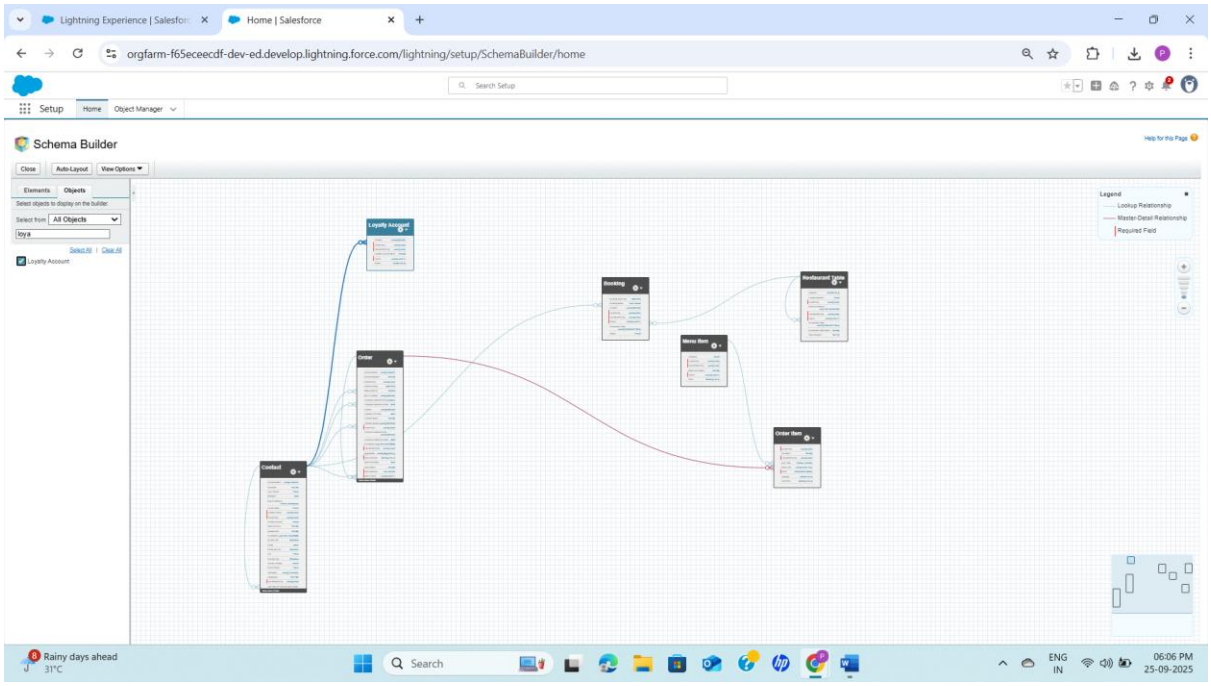
- **Core Role:** Stores all food/beverage orders.
- **Relationships:**
  - Master-Detail → **Order Item** (an order contains multiple items).
  - Lookup → **Contact** (who placed the order).
  - Lookup → **Booking** (for dine-in orders linked to reservations).
  - Lookup → **Loyalty Account** (if loyalty points are used).
- **Theory:**
  The **backbone of restaurant operations** — captures both dine-in and delivery orders. It ties customer transactions to billing, loyalty points, and reporting.

Order Item

- **Core Role:** Captures individual food items within an order.
- **Relationships:**
  - Master-Detail → **Order** (parent).
  - Lookup → **Menu Item** (what was ordered).
- **Theory:**
  Provides granularity. Instead of storing only the order total, the restaurant can track exactly which dishes were sold, in what quantity, and at what price.

Menu Item

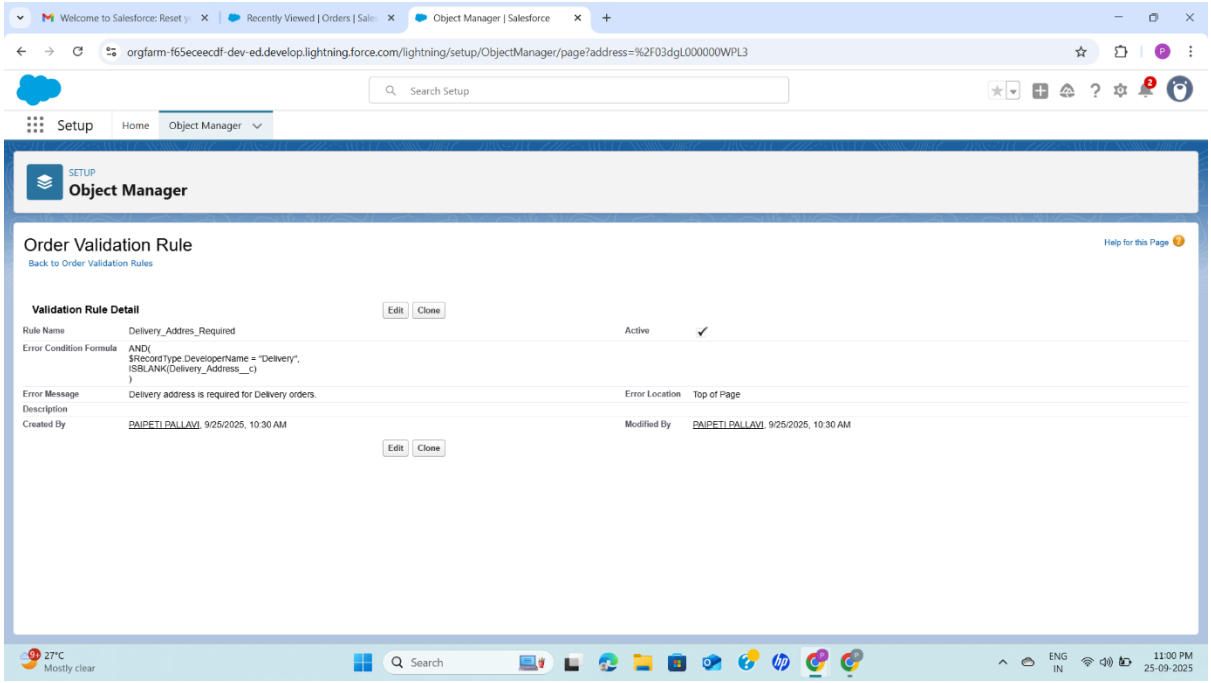
- **Core Role:** Master catalog of dishes.
- **Relationship:** Referenced by **Order Item**.
- **Theory:**
  The restaurant's product database. Every order item references a menu item, ensuring consistency in pricing and reporting.


Lightning Experience | Salesforce
Home | Salesforce
orgfarm-f65eceecdf-dev-ed.develop.lightning.force.com/lightning/setup/SchemaBuilder/home
Setup
Home
Object Manager
Schema Builder
Close
Auto-Layout
View Options
Elements
Objects
Select objects to display on the builder.
Select from
All Objects
loya
Select All
Clear All
Loyalty Account
Legend
Lookup Relationship
Master-Detail Relationship
Required Field
Loyalty Account
Booking
Restaurant Table
Order
Menu Item
Order Item
Contact
Rainy days ahead
31°C
Search
ENG
IN
06:06 PM
25-09-2025

PHASE 4

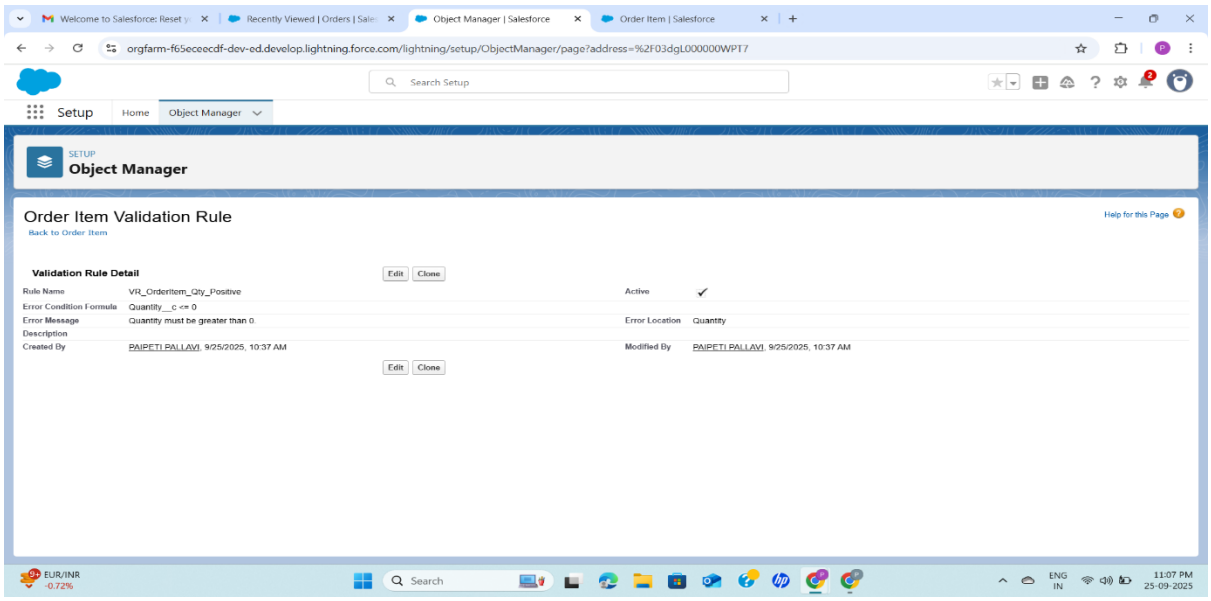
A. Delivery address required

1. Setup → **Object Manager** → **Order__c** → **Validation Rules** → **New**.
2. Fill:
    - **Rule Name:** Delivery_Address_Required
    - **Error Condition Formula:**
    - AND(
    - $RecordType.DeveloperName = "Delivery",
    - ISBLANK(Delivery_Address__c)
    -)
    - **Error Message:** Delivery address is required for Delivery orders.
    - **Error Location:** Field → Delivery_Address__c
3. Click **Save**, then **Activate**.
4. **Test:** Create/update an Order record with Record Type = Delivery and no address → expect error.



B. Quantity must be > 0

1. Setup → **Object Manager** → **Order_Item__c** → **Validation Rules** → **New**.
2. Fill:
    - **Rule Name:** Quantity_Positive
    - **Error Condition Formula:**
    - Quantity__c <= 0
    - **Error Message:** Quantity must be greater than 0.
    - **Error Location:** Field → Quantity__c
3. Save & Activate.
4. **Test:** Create Order Item with Quantity = 0 or negative → expect error.



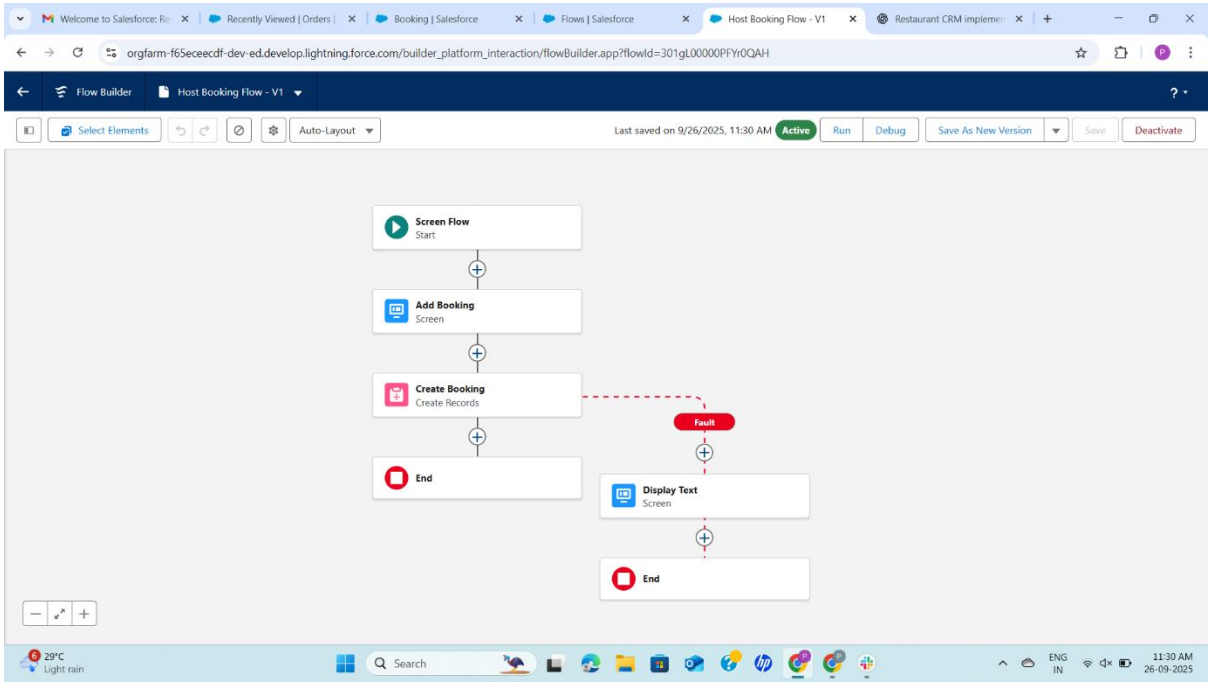
Screen Flow (Host Booking) — step-by-step

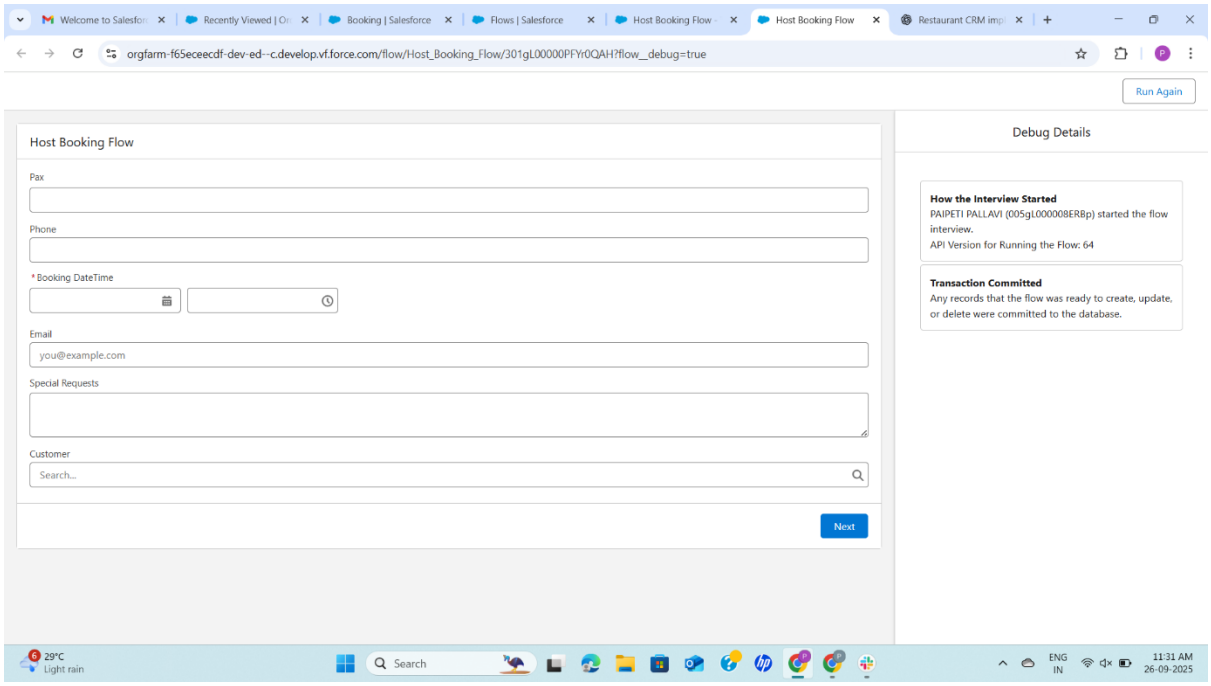
Goal: tablet-host enters booking, flow creates Booking__c, assigns table (auto or manual), notifies manager.

1. Setup → **Flow** → **New Flow** → choose **Screen Flow** → **Create**.
2. On canvas: + → **Screen**.
    - Label: Add Booking
    - Add components:
        - **Lookup** (Customer) → API/field: Customer__c (set Required)
        - **Number** (Pax) → Pax__c (Required)
        - **Date/Time** → Booking_DateTime__c (Required)
        - **Text Area** → Special_Requests__c
        - **Phone** & **Email** fields if needed
    - Save screen.
3. **Create Records** element:
    - Label: Create Booking Record
    - Create one Booking__c. Map screen fields to Booking__c fields (Customer__c ← {!Customer}, etc.).
4. **Decision** element: Table Available?
    - Option A: If you have a Table__c object, add a **Get Records** to find a table with Capacity__c >= {!Pax} and Is_Available__c = TRUE. If one found → route to Auto Assign.
    - Option B: If complex logic required, call an **Apex Invocable** (add **Action** → the invocable class) which returns Table Id.
5. **Screen** Confirm:
    - Show assigned table or a record choice to pick table manually. (Use Record Choice Set or Choice component bound to Table__c results.)

6. **Action(s)**:
    - Add **Create Task** or **Send Custom Notification** (Action → Send Custom Notification) for manager.
    - Or use **Action → Email Alert** (if you created an Email Alert) to send email.
7. Save → **Debug** with sample values → fix mapping.
8. Save → **Activate**.
9. Expose Flow:
    - Add to Lightning App page (App Builder) or create a Quick Action that launches the flow (Object → Buttons, Links, Actions → New Action → Flow). For tablet, add to the app page or make it full-screen.

Testing: on a device, run flow, confirm Booking__c created, table assigned, notifications received.





Record-Triggered Flow

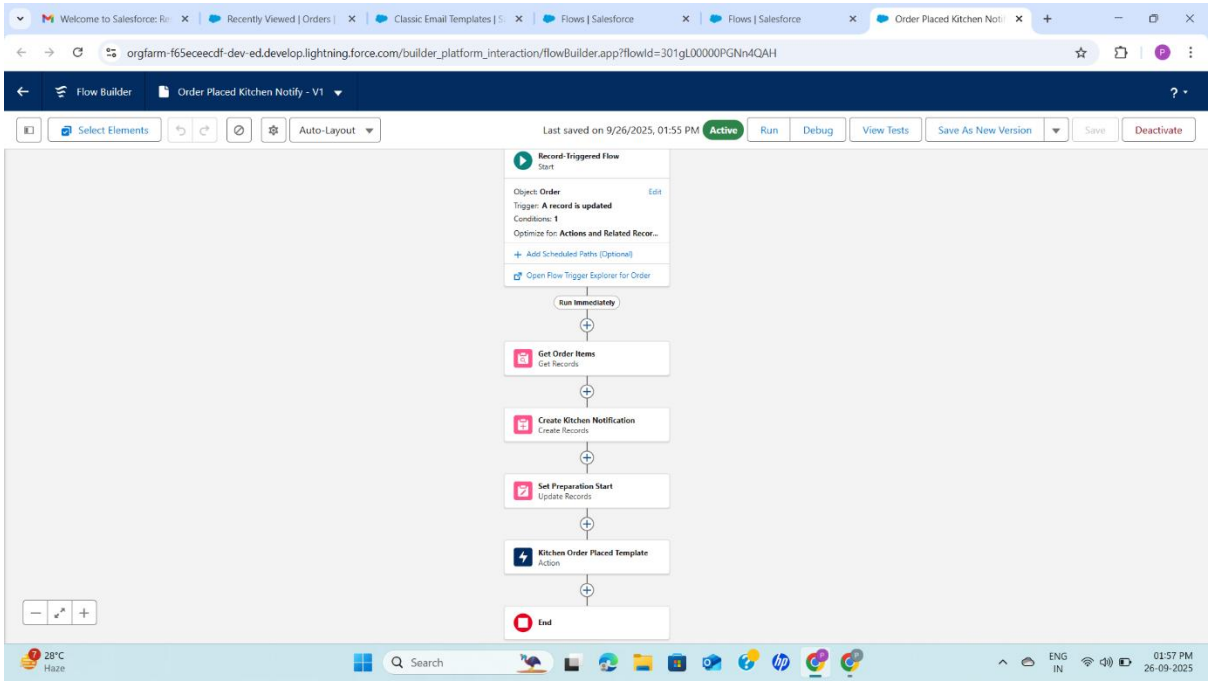
Goal: When Order__c.Status__c becomes **Placed**, notify kitchen, set Preparation_Start__c.

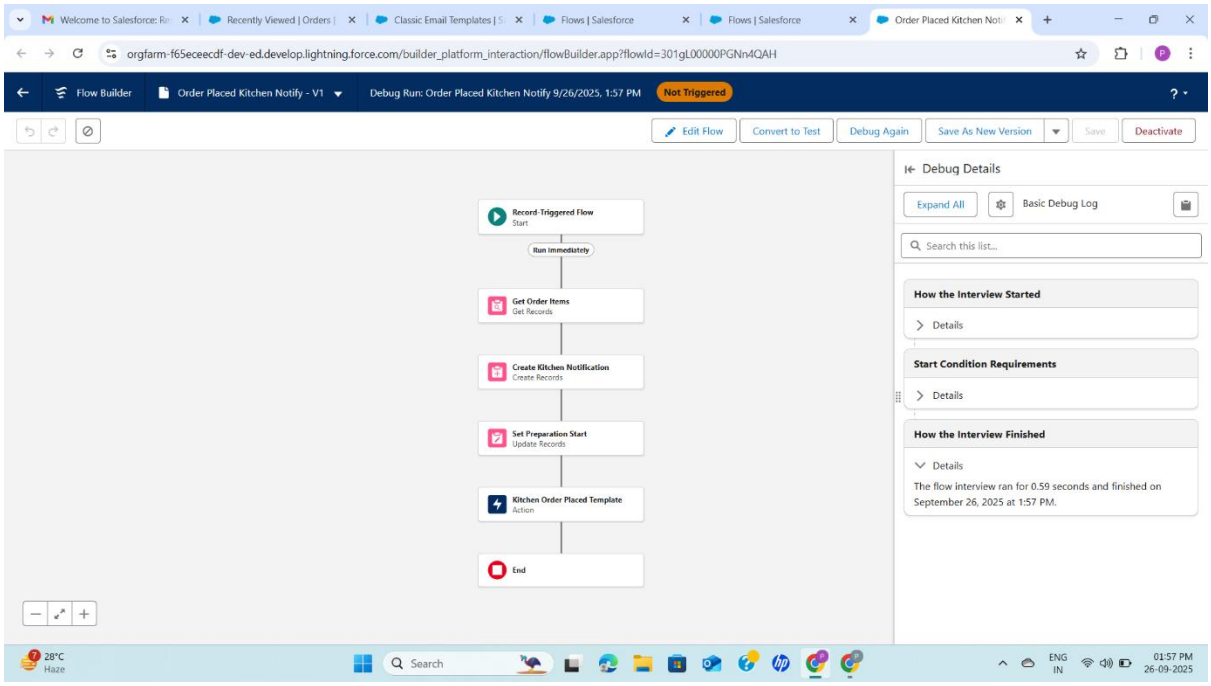
1. Setup → **Flows** → **New Flow** → **Record-Triggered Flow** → **Create**.
2. Configure start:
    - **Object:** Order__c
    - **Trigger:** A record is **updated**
    - **Condition Requirements:** Status__c EQUALS Placed (use picklist equals).
    - **When to Run: After the record is saved** (we will create other records & notifications).
    - **Optimize the Flow for:** Actions and Related Records.
    - Click **Done**.
3. First **Get Records**:
    - Label: Get Order Items
    - Object: Order_Item__c
    - Filter: Order__c = $Record.Id
    - Get all records, store as collection.
4. (Optional) **Decision**: If no order items → end (avoid empty operations).
5. **Create Records**: create Kitchen_Order__c (or your custom notification record) — you can create one record per Order or one per item.
    - For large sets, create one summary Kitchen_Order and attach items via related list.
6. **Update Records**:
    - Label: Set Prep Start
    - Update record: choose the triggering record (Order__c where Id = $Record.Id) → set Preparation_Start__c = $Flow.CurrentDateTime (or use formula NOW()).
7. **Action**: Send Notification

- Option A: **Send Custom Notification** action → choose Notification Type (create one first) and set Recipient IDs (Queue/Users).
- Option B: Use **Action → Email Alert** (pre-created Email Alert).
- Option C: **Post to Chatter** action: Post to a group or user.

8. **Add Scheduled Path** (if you want follow-up/reminder or auto-cancel):
   - In the Start element, click + **Add Scheduled Paths**, set label Auto Cancel If Not Confirmed, offset 24 Hours After Trigger (or X hours), then on that scheduled path add a Decision: if Order__c.Confirmed__c = FALSE → **Update Records** to set Status__c = 'Cancelled' and send email.
9. Save → **Debug**: run with a test order using "Run" debug in flow (use record Id). Review Flow Interviews in Setup → Paused Flow Interviews (if scheduled).
10. Activate.

Testing: Update an Order to Placed → check Kitchen_Order created, Preparation_Start updated, notifications emailed.





Scheduled Flow

Goal: nightly job to process Loyalty_Account__c and update balances.

1. Setup → **Flows** → **New Flow** → **Scheduled-Triggered Flow** → **Create**.
2. Configure schedule:
    - **Start Date & Time**: pick next run time (e.g., tomorrow 02:00 AM)
    - **Frequency:** Daily
    - **Time Zone:** Org default (confirm timezone)
3. Canvas:
    - **Get Records**: Loyalty_Account__c → filter criteria as needed (e.g., Active = TRUE). **Store all records** into a collection.
    - **Loop** over the collection (Loop variable: loopLoyaltyAccounts).
        - In loop: **Decision** → has expiry? has activity?
        - If points to add: Add a record to a collection variable transactionsToCreate (build a record variable using Assignment).
        - Also use Assignment to compute new Points_Balance__c into a accountsToUpdate collection (create collection of records to update).
    - After Loop: **Create Records** (create many Loyalty_Transaction__c using transactionsToCreate) — this uses bulk create.
    - **Update Records** — use accountsToUpdate collection to bulk update balances.
4. Save → Debug (use a small test dataset) → Activate.
5. **Scale note:** If your org has thousands+ accounts, consider Batch Apex or break the scheduled flow to process smaller slices (e.g., criteria by last modified ranges) to avoid platform limits.

Create Notification Type (for in-app)

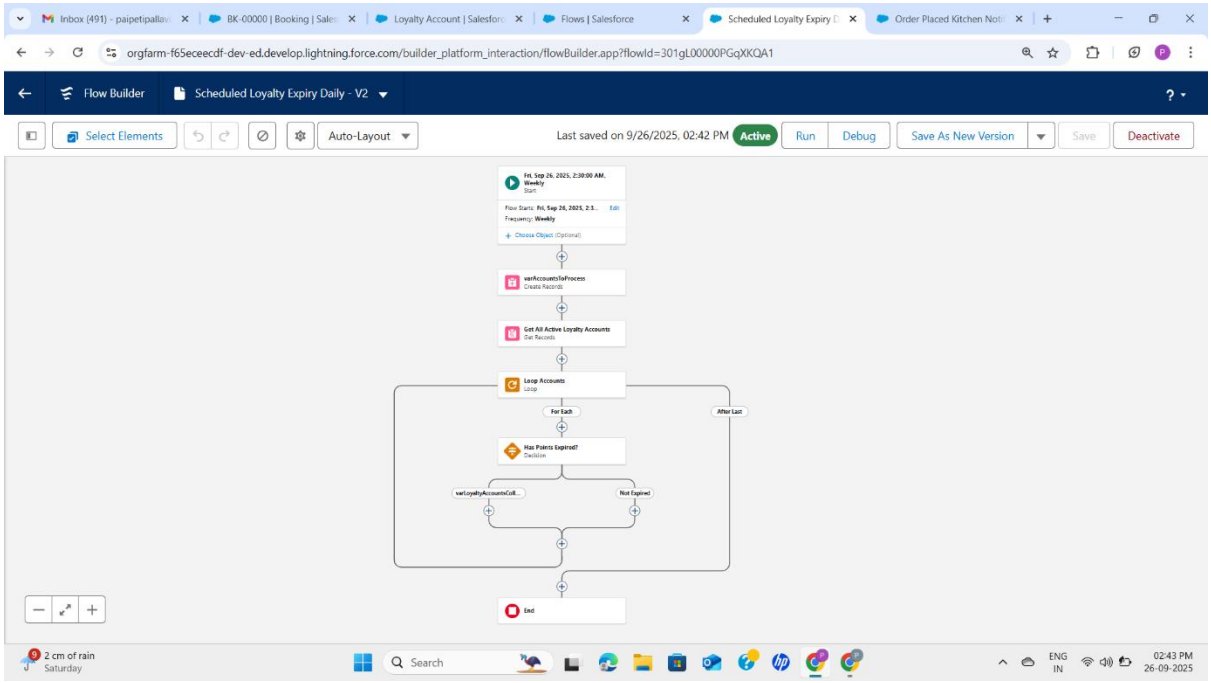
1. Setup → **Notification Builder** → **Notification Type** → **New**.
2. Label: Kitchen_Order_Alert, Channels: In-App and Desktop if needed. Save.

Create Email Template & Email Alert

1. Setup → **Email Templates** → **New Email Template** (Lightning) → create template using merge fields.
2. Setup → **Email Alerts** → **New Email Alert**:
    - Object: Order__c
    - Email Template: choose above
    - Recipients: User, Queue, Related Users (e.g., Order.Owner)
3. Use this Email Alert in Flow:
    - In Flow canvas: **Action** → search for **Email Alert** → choose your alert.

Send Custom Notification in Flow

1. In Flow canvas: **Action** → Send Custom Notification → choose Notification Type created earlier.
2. Supply Title, Body, Recipient Id(s) (use {!User.Id} or collection).



✅ Created a Booking record (Booking__c object)

- Record Name auto-generated: **BK-00000** (probably an Auto-Number field).

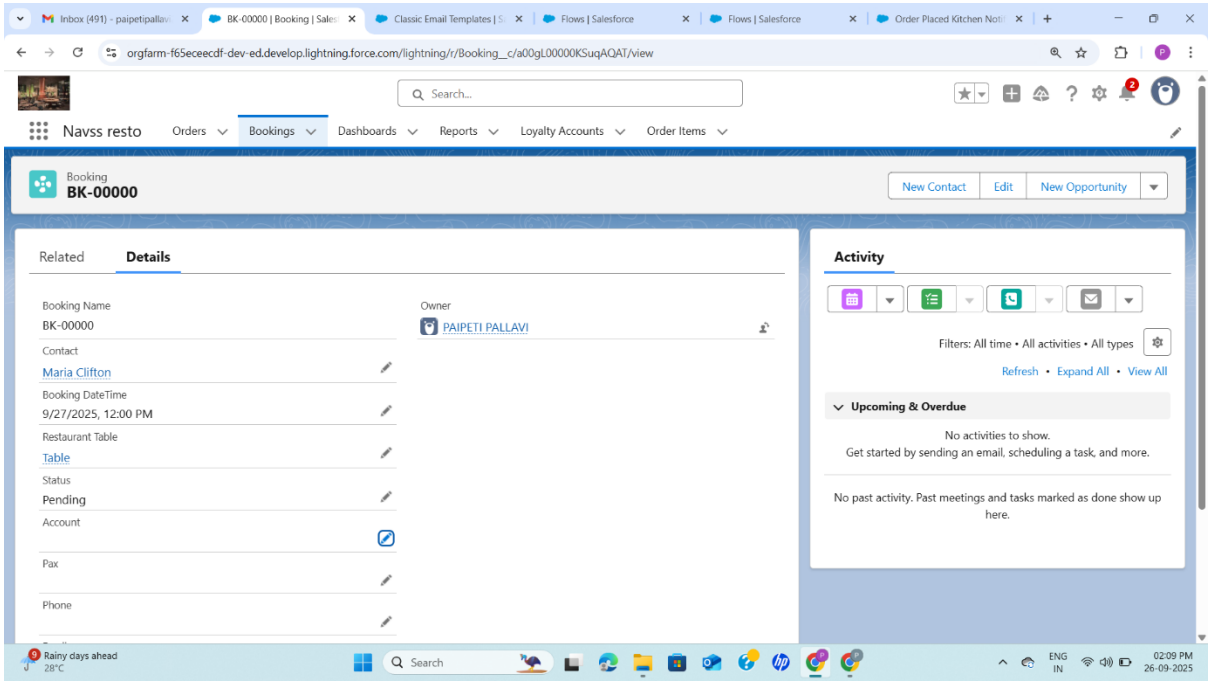
✅ Filled core Booking fields:

- **Contact** → Selected *Maria Clifton* (lookup to Contact).
- **Booking DateTime** → 9/27/2025, 12:00 PM.
- **Restaurant Table** → Linked to a Table record (lookup).
- **Status** → Set as *Pending* (picklist).
- **Account** → Left empty (lookup to Account).
- **Pax** → Empty (should store number of guests).
- **Phone, Email, Special Requests** → Empty (but available for input).

 Record Owner → You (Paipeti Pallavi).

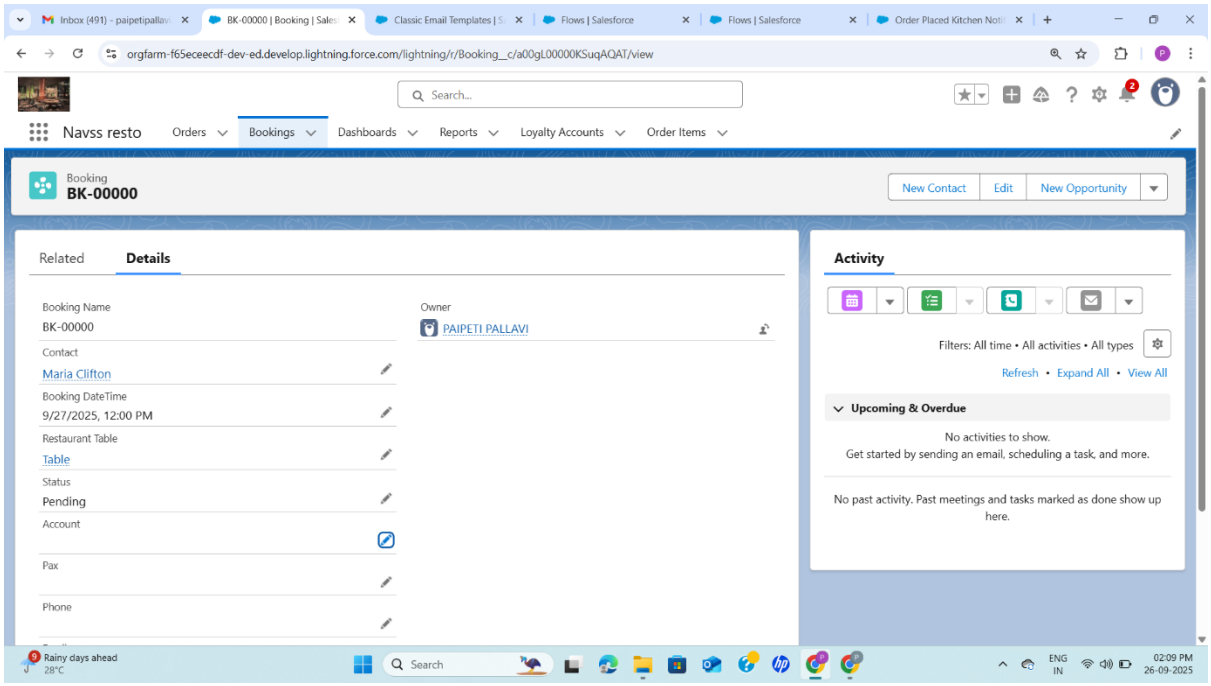
- This means you are responsible for the booking.

 Activity Panel → No tasks/emails/events logged yet.



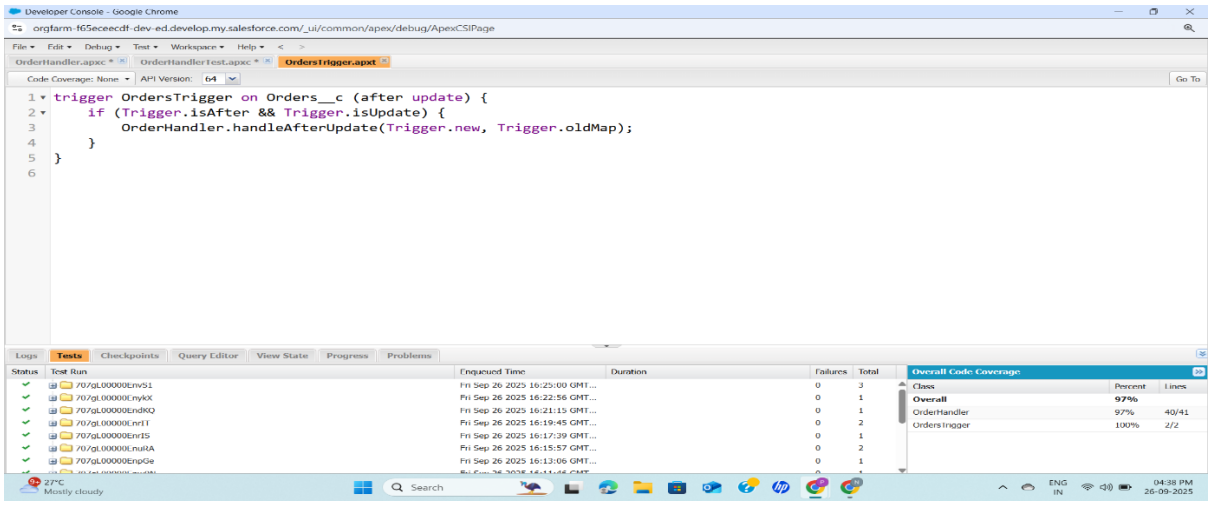
PHASE 5

Before running the apex classes the status is in pending state

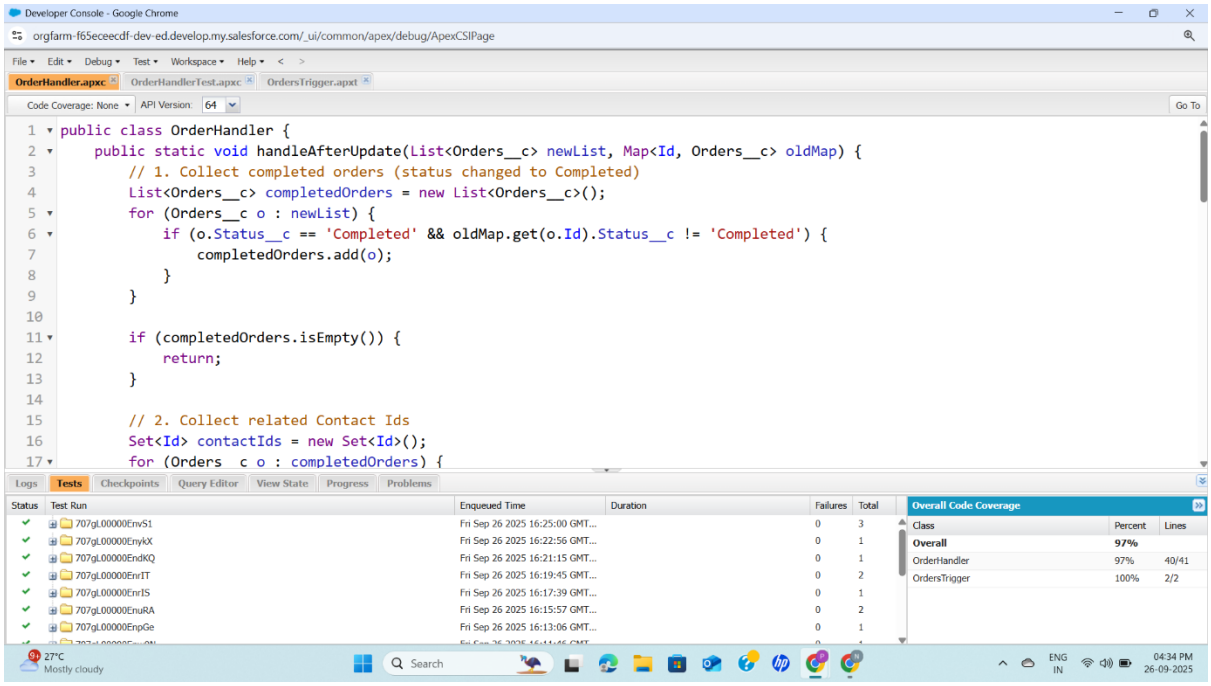


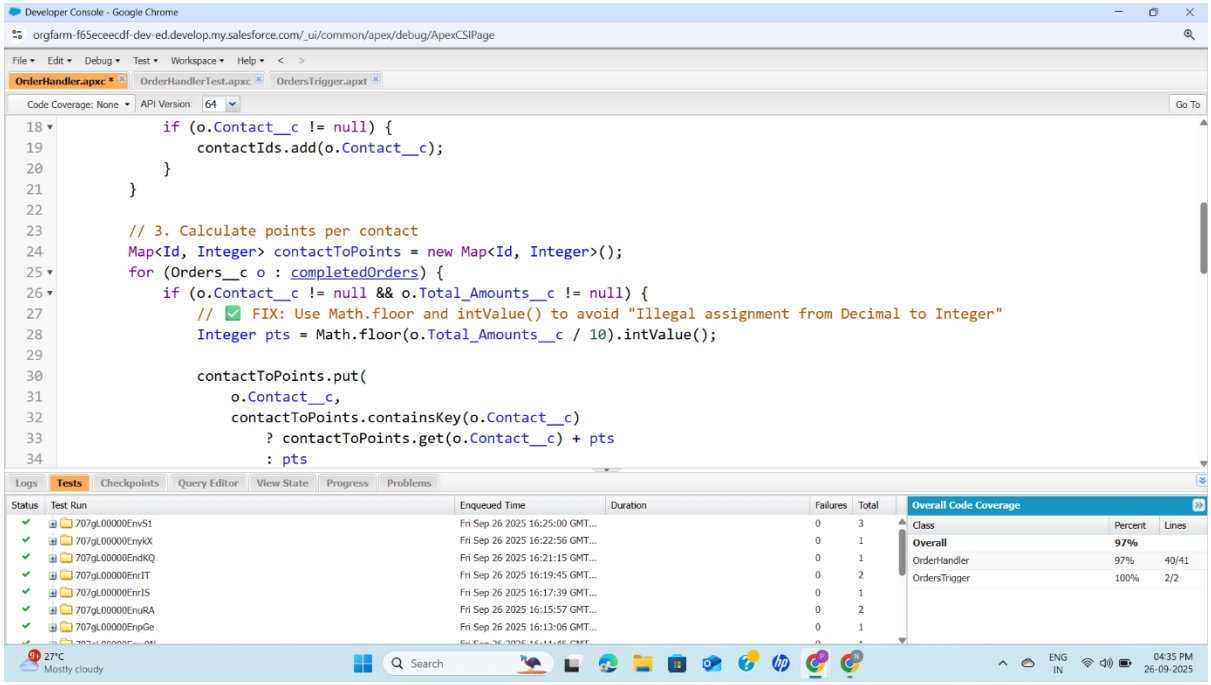
I have used the apex to change the status to the confirmed state.

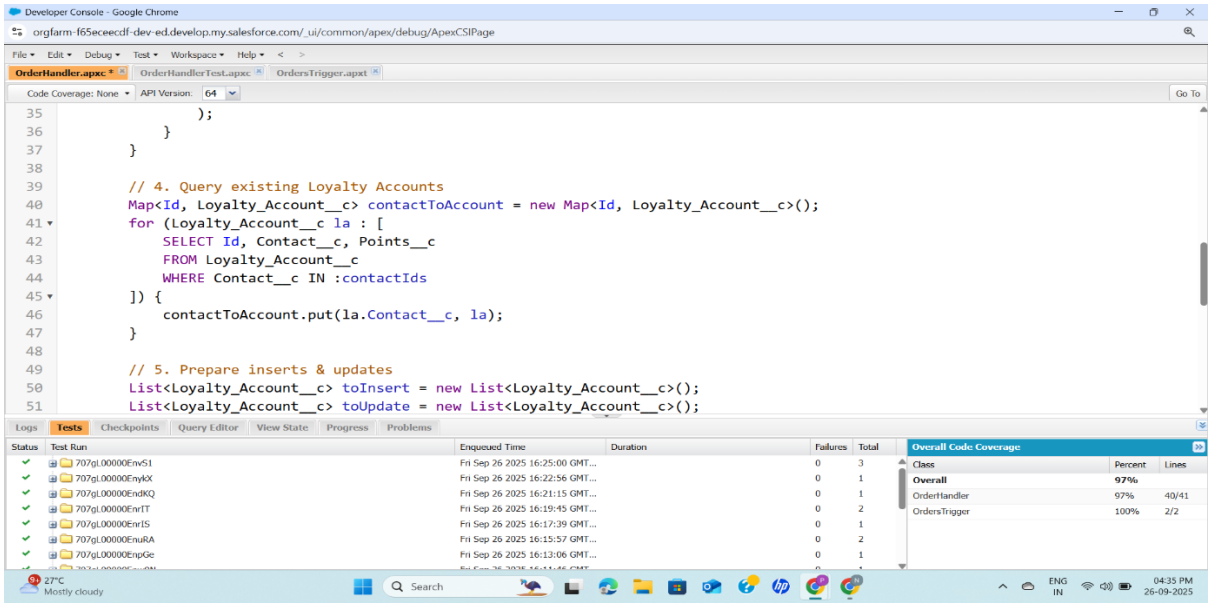
OrderTrigger.trigger

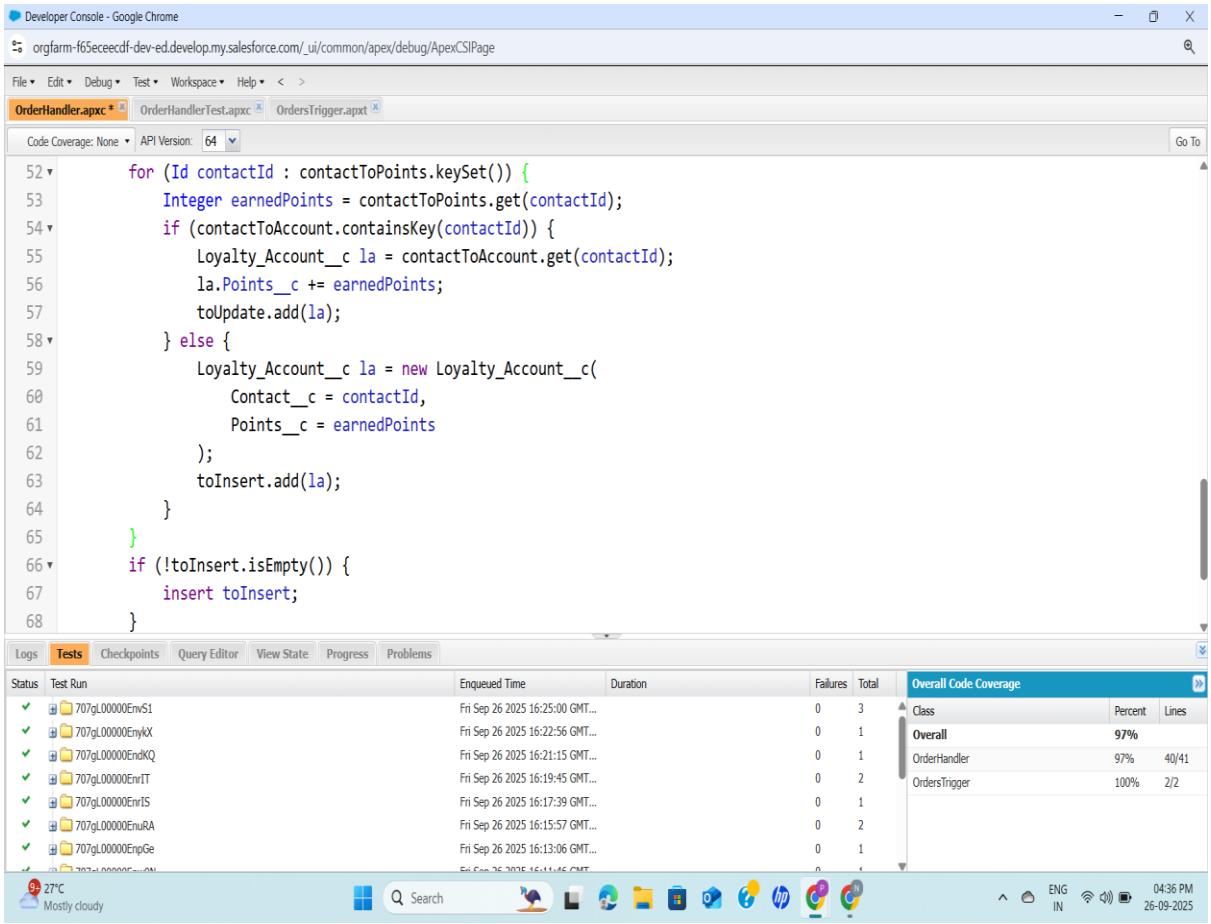


OrderHandler.apxc

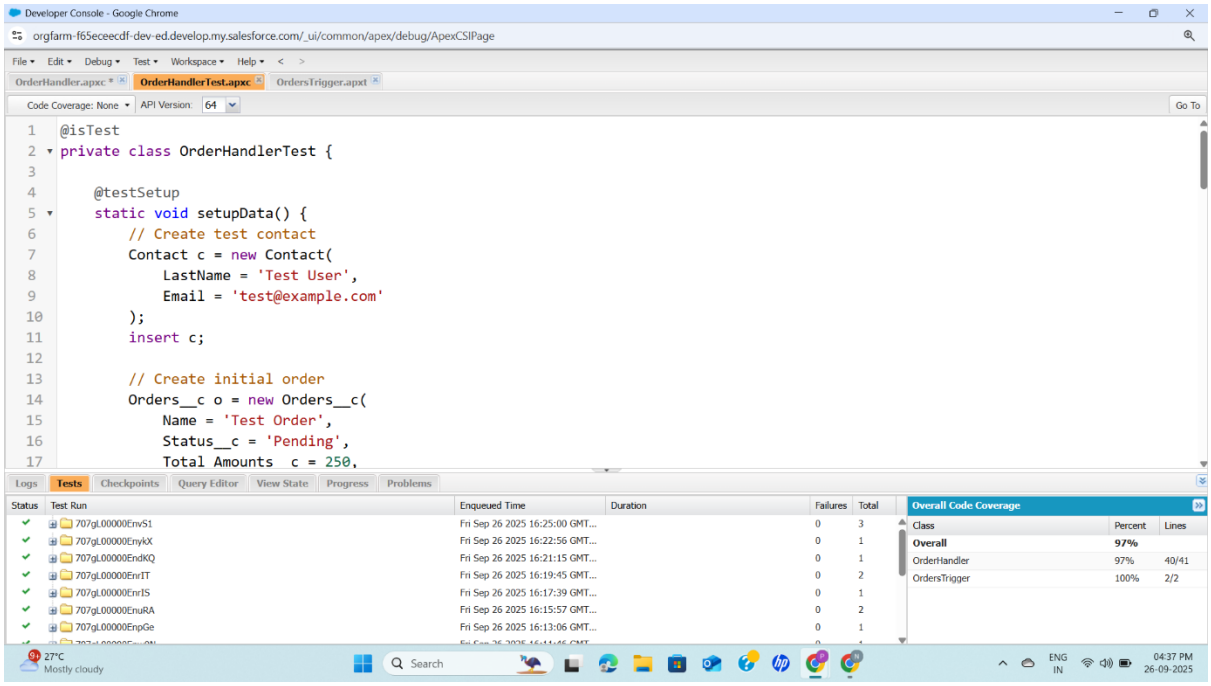

Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHandler.apxc
OrderHandlerTest.apxc
OrdersTrigger.apxt
Code Coverage: None
API Version: 64
Go To
1 public class OrderHandler {
2 public static void handleAfterUpdate(List<Orders__c> newList, Map<Id, Orders__c> oldMap) {
3 // 1. Collect completed orders (status changed to Completed)
4 List<Orders__c> completedOrders = new List<Orders__c>();
5 for (Orders__c o : newList) {
6 if (o.Status__c == 'Completed' && oldMap.get(o.Id).Status__c != 'Completed') {
7 completedOrders.add(o);
8 }
9 }
10
11 if (completedOrders.isEmpty()) {
12 return;
13 }
14
15 // 2. Collect related Contact Ids
16 Set<Id> contactIds = new Set<Id>();
17 for (Orders c o : completedOrders) {
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Status
Test Run
Enqueued Time
Duration
Failures
Total
707gL00000EnvS1 Fri Sep 26 2025 16:25:00 GMT... 0 3
707gL00000EnykX Fri Sep 26 2025 16:22:56 GMT... 0 1
707gL00000EndKQ Fri Sep 26 2025 16:21:15 GMT... 0 1
707gL00000EnrIT Fri Sep 26 2025 16:19:45 GMT... 0 2
707gL00000EnrIS Fri Sep 26 2025 16:17:39 GMT... 0 1
707gL00000EnuRA Fri Sep 26 2025 16:15:57 GMT... 0 2
707gL00000EnpGe Fri Sep 26 2025 16:13:06 GMT... 0 1
Overall Code Coverage
Class
Percent
Lines
Overall 97%
OrderHandler 97% 40/41
OrdersTrigger 100% 2/2
27°C
Mostly cloudy
Search
ENG
IN
04:34 PM
26-09-2025

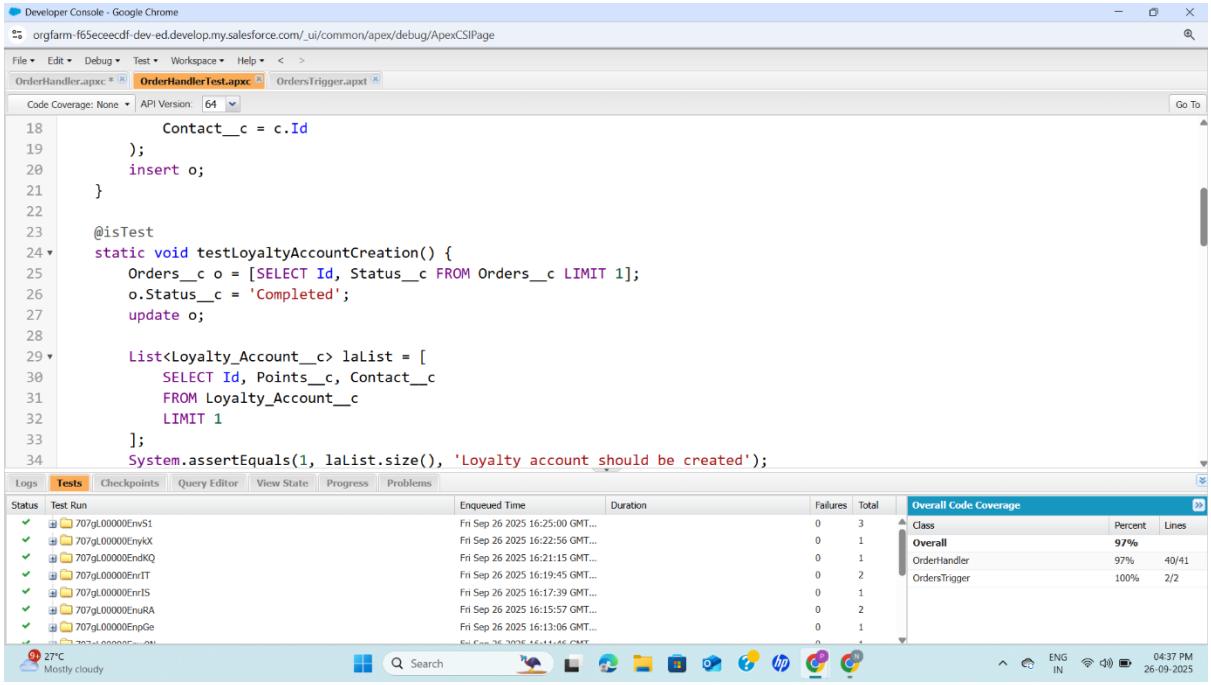

Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHandler.apxc *
OrderHandlerTest.apxc
OrdersTrigger.apxt
Code Coverage: None
API Version: 64
Go To
18 if (o.Contact__c != null) {
19 contactIds.add(o.Contact__c);
20 }
21 }
22
23 // 3. Calculate points per contact
24 Map<Id, Integer> contactToPoints = new Map<Id, Integer>();
25 for (Orders__c o : completedOrders) {
26 if (o.Contact__c != null && o.Total_Amounts__c != null) {
27 // ✅ FIX: Use Math.floor and intValue() to avoid "Illegal assignment from Decimal to Integer"
28 Integer pts = Math.floor(o.Total_Amounts__c / 10).intValue();
29
30 contactToPoints.put(
31 o.Contact__c,
32 contactToPoints.containsKey(o.Contact__c)
33 ? contactToPoints.get(o.Contact__c) + pts
34 : pts
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Status
Test Run
Enqueued Time
Duration
Failures
Total
707gL00000EnvS1 Fri Sep 26 2025 16:25:00 GMT... 0 3
707gL00000EnykX Fri Sep 26 2025 16:22:56 GMT... 0 1
707gL00000EndKQ Fri Sep 26 2025 16:21:15 GMT... 0 1
707gL00000EnrIT Fri Sep 26 2025 16:19:45 GMT... 0 2
707gL00000EnrIS Fri Sep 26 2025 16:17:39 GMT... 0 1
707gL00000EnuRA Fri Sep 26 2025 16:15:57 GMT... 0 2
707gL00000EnpGe Fri Sep 26 2025 16:13:06 GMT... 0 1
Overall Code Coverage
Class
Percent
Lines
Overall 97%
OrderHandler 97% 40/41
OrdersTrigger 100% 2/2
27°C
Mostly cloudy
Search
ENG
IN
04:35 PM
26-09-2025

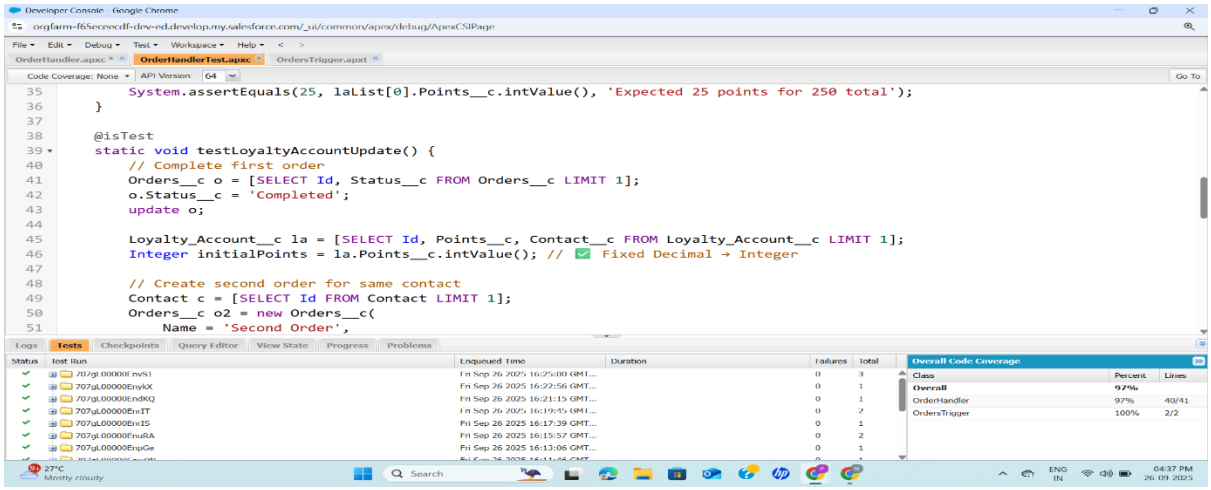


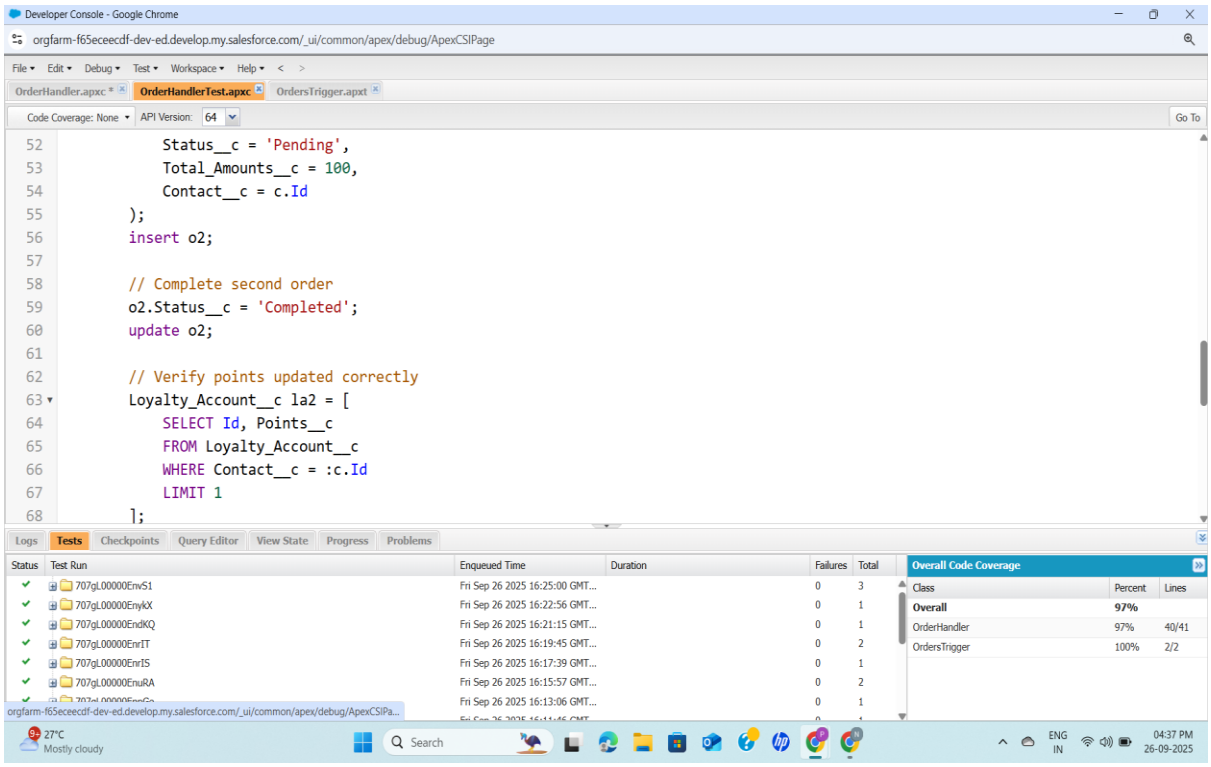


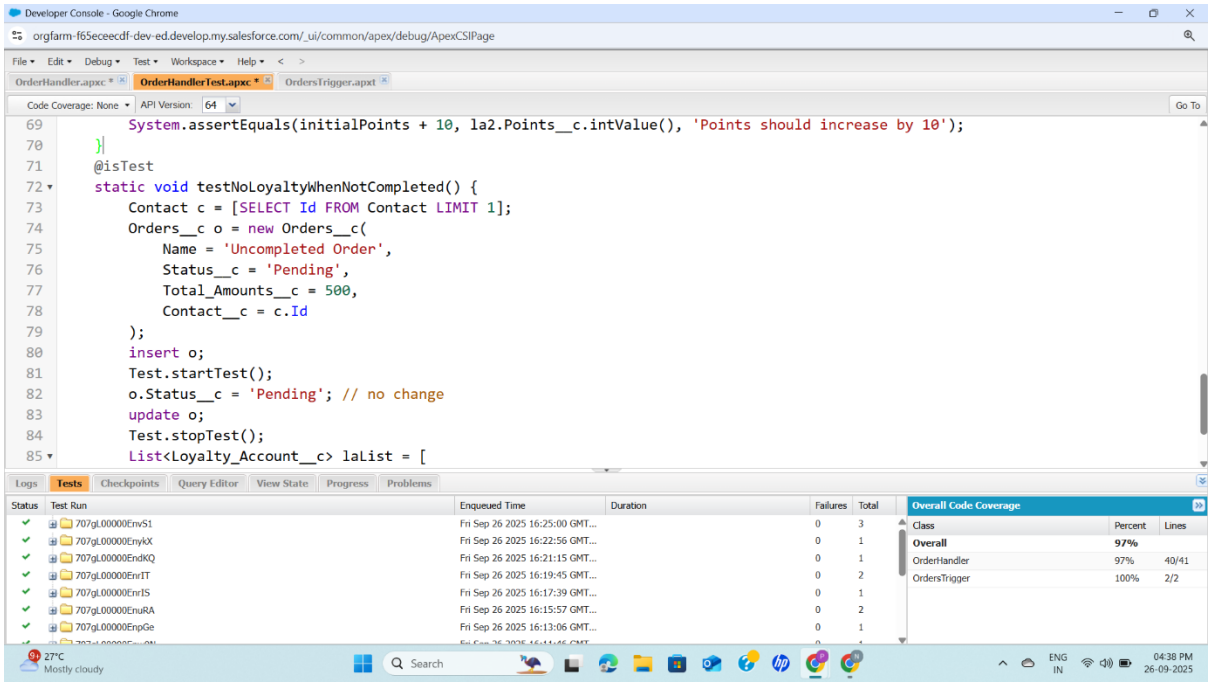
OrderHandlerTest.apxc

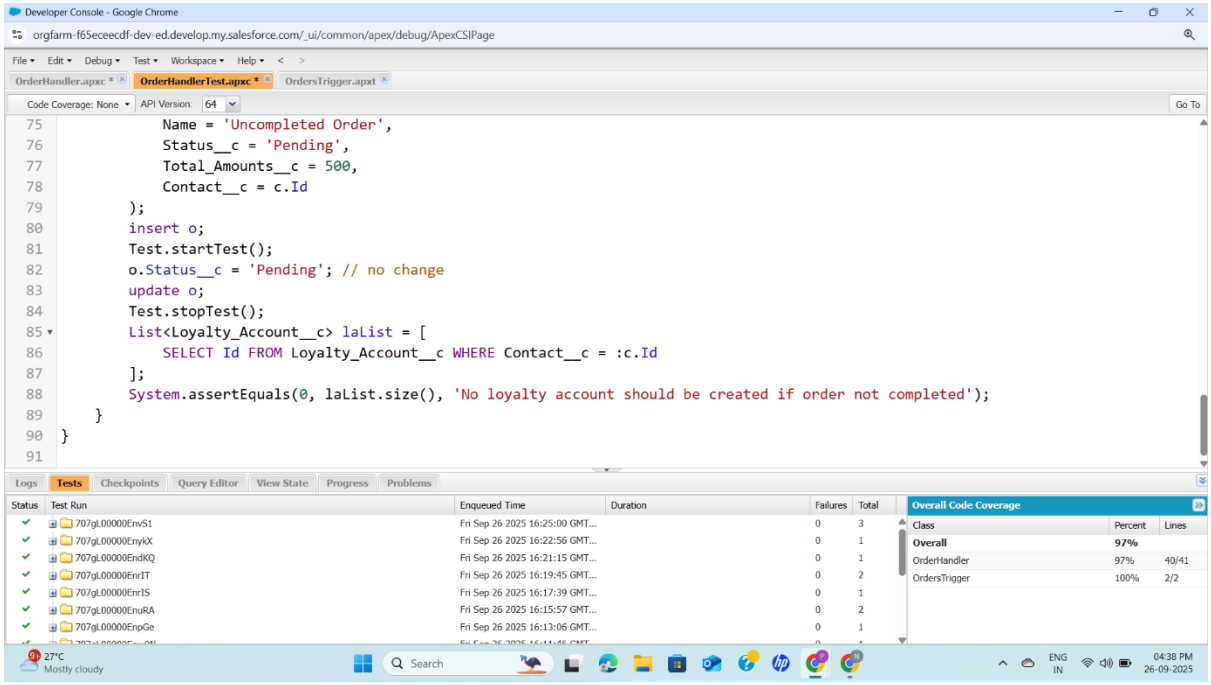

Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHandler.apxc
OrderHandlerTest.apxc
OrdersTrigger.apxt
Code Coverage: None
API Version: 64
Go To
@isTest
private class OrderHandlerTest {
@testSetup
static void setupData() {
// Create test contact
Contact c = new Contact(
LastName = 'Test User',
Email = 'test@example.com'
);
insert c;
// Create initial order
Orders__c o = new Orders__c(
Name = 'Test Order',
Status__c = 'Pending',
Total Amounts c = 250,
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Status
Test Run
Enqueued Time
Duration
Failures
Total
707gL00000EnvS1 Fri Sep 26 2025 16:25:00 GMT... 0 3
707gL00000EnykX Fri Sep 26 2025 16:22:56 GMT... 0 1
707gL00000EndKQ Fri Sep 26 2025 16:21:15 GMT... 0 1
707gL00000EnrIT Fri Sep 26 2025 16:19:45 GMT... 0 2
707gL00000EnrIS Fri Sep 26 2025 16:17:39 GMT... 0 1
707gL00000EnuRA Fri Sep 26 2025 16:15:57 GMT... 0 2
707gL00000EnpGe Fri Sep 26 2025 16:13:06 GMT... 0 1
Overall Code Coverage
Class
Percent
Lines
Overall 97%
OrderHandler 97% 40/41
OrdersTrigger 100% 2/2
27°C
Mostly cloudy
Search
ENG
IN
04:37 PM
26-09-2025


Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHandler.apxc
OrderHandlerTest.apxc
OrdersTrigger.apxt
Code Coverage: None
API Version: 64
Go To
Contact__c = c.Id
);
insert o;
}
@isTest
static void testLoyaltyAccountCreation() {
Orders__c o = [SELECT Id, Status__c FROM Orders__c LIMIT 1];
o.Status__c = 'Completed';
update o;
List<Loyalty_Account__c> laList = [
SELECT Id, Points__c, Contact__c
FROM Loyalty_Account__c
LIMIT 1
];
System.assertEquals(1, laList.size(), 'Loyalty account should be created');
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Status
Test Run
Enqueued Time
Duration
Failures
Total
707gL00000EnvS1 Fri Sep 26 2025 16:25:00 GMT... 0 3
707gL00000EnykX Fri Sep 26 2025 16:22:56 GMT... 0 1
707gL00000EndKQ Fri Sep 26 2025 16:21:15 GMT... 0 1
707gL00000EnrIT Fri Sep 26 2025 16:19:45 GMT... 0 2
707gL00000EnrIS Fri Sep 26 2025 16:17:39 GMT... 0 1
707gL00000EnuRA Fri Sep 26 2025 16:15:57 GMT... 0 2
707gL00000EnpGe Fri Sep 26 2025 16:13:06 GMT... 0 1
Overall Code Coverage
Class
Percent
Lines
Overall 97%
OrderHandler 97% 40/41
OrdersTrigger 100% 2/2
27°C
Mostly cloudy
Search
ENG
IN
04:37 PM
26-09-2025

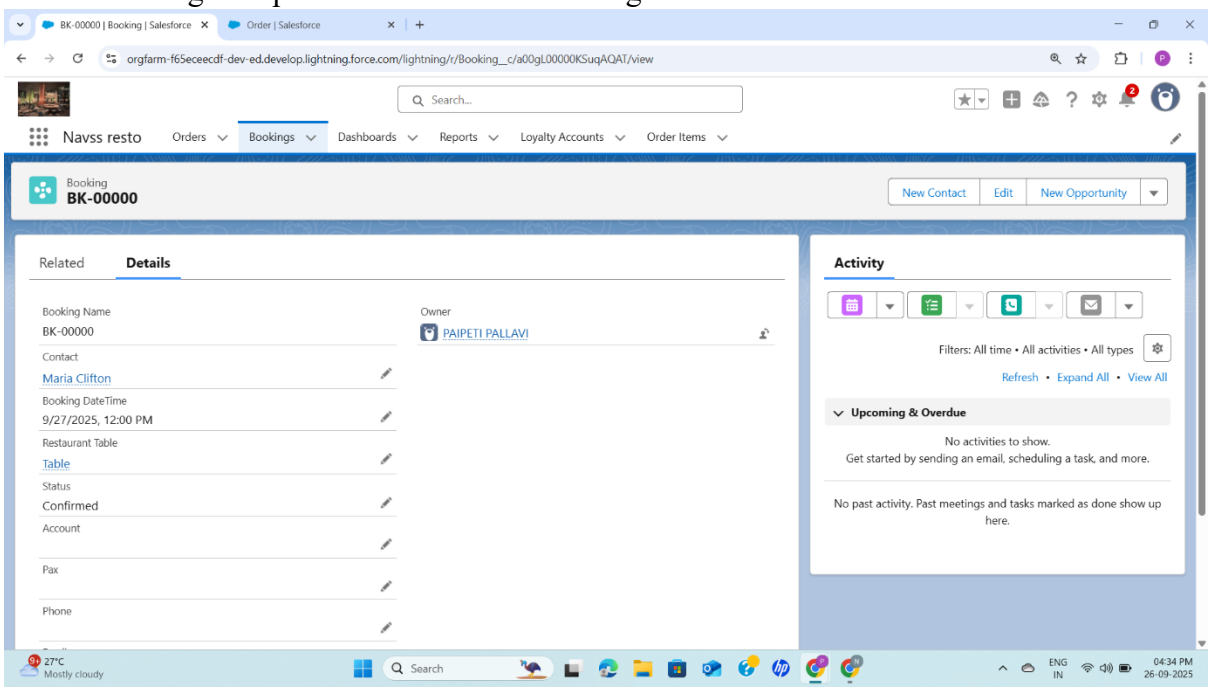

Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
OrderHandler.apxc
OrderHandlerTest.apxc
OrdersTrigger.apxt
System.assertEquals(25, laList[0].Points__c.intValue(), 'Expected 25 points for 250 total');
}
@isTest
static void testLoyaltyAccountUpdate() {
// Complete first order
Orders__c o = [SELECT Id, Status__c FROM Orders__c LIMIT 1];
o.Status__c = 'Completed';
update o;
Loyalty_Account__c la = [SELECT Id, Points__c, Contact__c FROM Loyalty_Account__c LIMIT 1];
Integer initialPoints = la.Points__c.intValue(); // Fixed Decimal → Integer
// Create second order for same contact
Contact c = [SELECT Id FROM Contact LIMIT 1];
Orders__c o2 = new Orders__c(
Name = 'Second Order',
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Status
Test Run
Enqueued Time
Duration
Failures
Total
Overall Code Coverage
Class
Percent
Lines
Overall
97%
OrderHandler
97%
40/41
OrdersTrigger
100%
2/2
27°C
Mostly cloudy
Search
ENG
IN
04:37 PM
26-09-2025


Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
OrderHandler.apxc
OrderHandlerTest.apxc
OrdersTrigger.apxt
Status__c = 'Pending',
Total_Amounts__c = 100,
Contact__c = c.Id
);
insert o2;
// Complete second order
o2.Status__c = 'Completed';
update o2;
// Verify points updated correctly
Loyalty_Account__c la2 = [
SELECT Id, Points__c
FROM Loyalty_Account__c
WHERE Contact__c = :c.Id
LIMIT 1
];
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Status
Test Run
Enqueued Time
Duration
Failures
Total
Overall Code Coverage
Class
Percent
Lines
Overall
97%
OrderHandler
97%
40/41
OrdersTrigger
100%
2/2
27°C
Mostly cloudy
Search
ENG
IN
04:37 PM
26-09-2025


Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHandler.apxc * OrderHandlerTest.apxc * OrdersTrigger.apxt
Code Coverage: None API Version: 64 Go To
69 System.assertEquals(initialPoints + 10, la2.Points__c.intValue(), 'Points should increase by 10');
70 }
71 @isTest
72 static void testNoLoyaltyWhenNotCompleted() {
73 Contact c = [SELECT Id FROM Contact LIMIT 1];
74 Orders__c o = new Orders__c(
75 Name = 'Uncompleted Order',
76 Status__c = 'Pending',
77 Total_Amounts__c = 500,
78 Contact__c = c.Id
79);
80 insert o;
81 Test.startTest();
82 o.Status__c = 'Pending'; // no change
83 update o;
84 Test.stopTest();
85 List<Loyalty_Account__c> laList = [
Logs Tests Checkpoints Query Editor View State Progress Problems
Status Test Run Enqueued Time Duration Failures Total
707gL00000EnvS1 Fri Sep 26 2025 16:25:00 GMT... 0 3
707gL00000EnykX Fri Sep 26 2025 16:22:56 GMT... 0 1
707gL00000EndKQ Fri Sep 26 2025 16:21:15 GMT... 0 1
707gL00000EnrIT Fri Sep 26 2025 16:19:45 GMT... 0 2
707gL00000EnrIS Fri Sep 26 2025 16:17:39 GMT... 0 1
707gL00000EnuRA Fri Sep 26 2025 16:15:57 GMT... 0 2
707gL00000EnpGe Fri Sep 26 2025 16:13:06 GMT... 0 1
Overall Code Coverage
Class Percent Lines
Overall 97%
OrderHandler 97% 40/41
OrdersTrigger 100% 2/2
27°C Mostly cloudy
Search
ENG IN
04:38 PM 26-09-2025


Developer Console - Google Chrome
orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHandler.apxc * OrderHandlerTest.apxc * OrdersTrigger.apxt
Code Coverage: None API Version: 64 Go To
75 Name = 'Uncompleted Order',
76 Status__c = 'Pending',
77 Total_Amounts__c = 500,
78 Contact__c = c.Id
79);
80 insert o;
81 Test.startTest();
82 o.Status__c = 'Pending'; // no change
83 update o;
84 Test.stopTest();
85 List<Loyalty_Account__c> laList = [
86 SELECT Id FROM Loyalty_Account__c WHERE Contact__c = :c.Id
87];
88 System.assertEquals(0, laList.size(), 'No loyalty account should be created if order not completed');
89 }
90 }
91
Logs Tests Checkpoints Query Editor View State Progress Problems
Status Test Run Enqueued Time Duration Failures Total
707gL00000EnvS1 Fri Sep 26 2025 16:25:00 GMT... 0 3
707gL00000EnykX Fri Sep 26 2025 16:22:56 GMT... 0 1
707gL00000EndKQ Fri Sep 26 2025 16:21:15 GMT... 0 1
707gL00000EnrIT Fri Sep 26 2025 16:19:45 GMT... 0 2
707gL00000EnrIS Fri Sep 26 2025 16:17:39 GMT... 0 1
707gL00000EnuRA Fri Sep 26 2025 16:15:57 GMT... 0 2
707gL00000EnpGe Fri Sep 26 2025 16:13:06 GMT... 0 1
Overall Code Coverage
Class Percent Lines
Overall 97%
OrderHandler 97% 40/41
OrdersTrigger 100% 2/2
27°C Mostly cloudy
Search
ENG IN
04:38 PM 26-09-2025

After executing the apex codes the status is changed to the confirmed



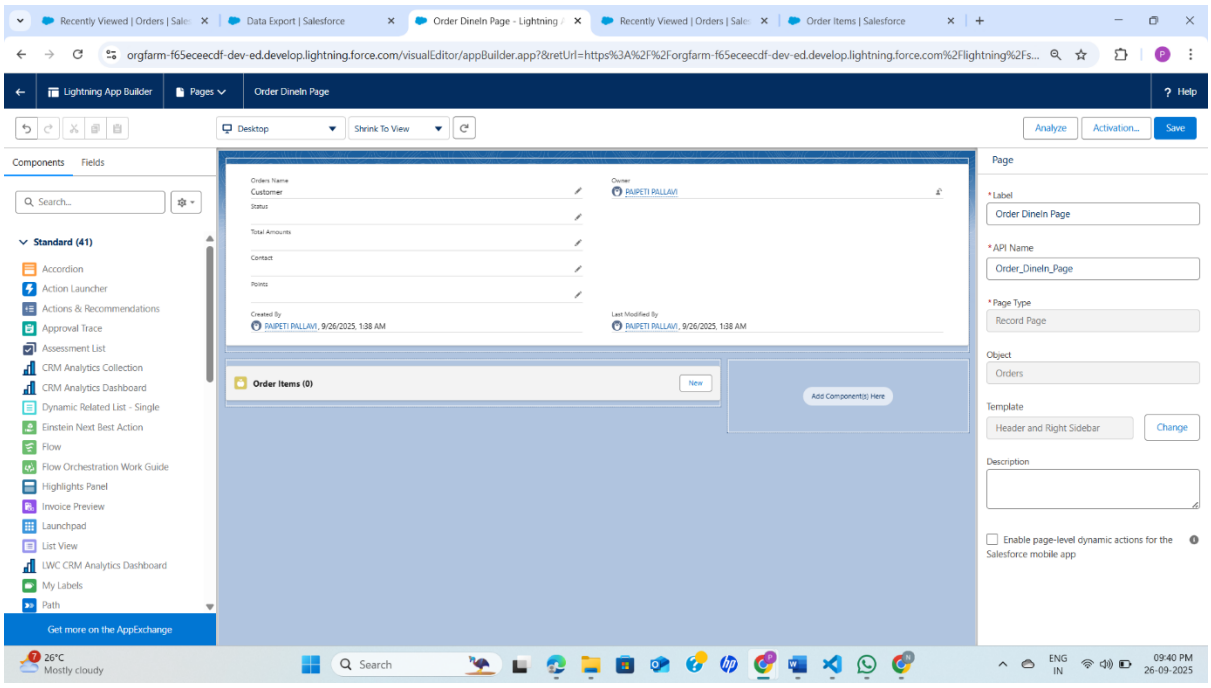
PHASE 6

LIGHTINING WEB COMPONENTS

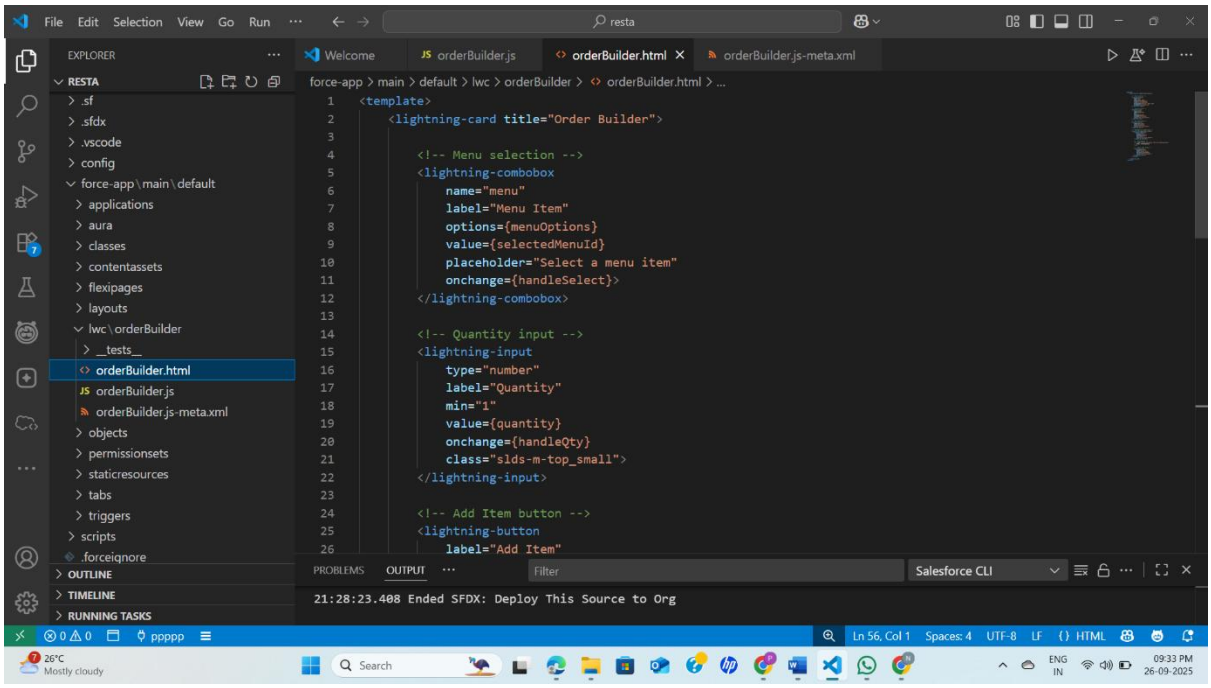
Before creating the LWC components I have created the created the lightning app builder

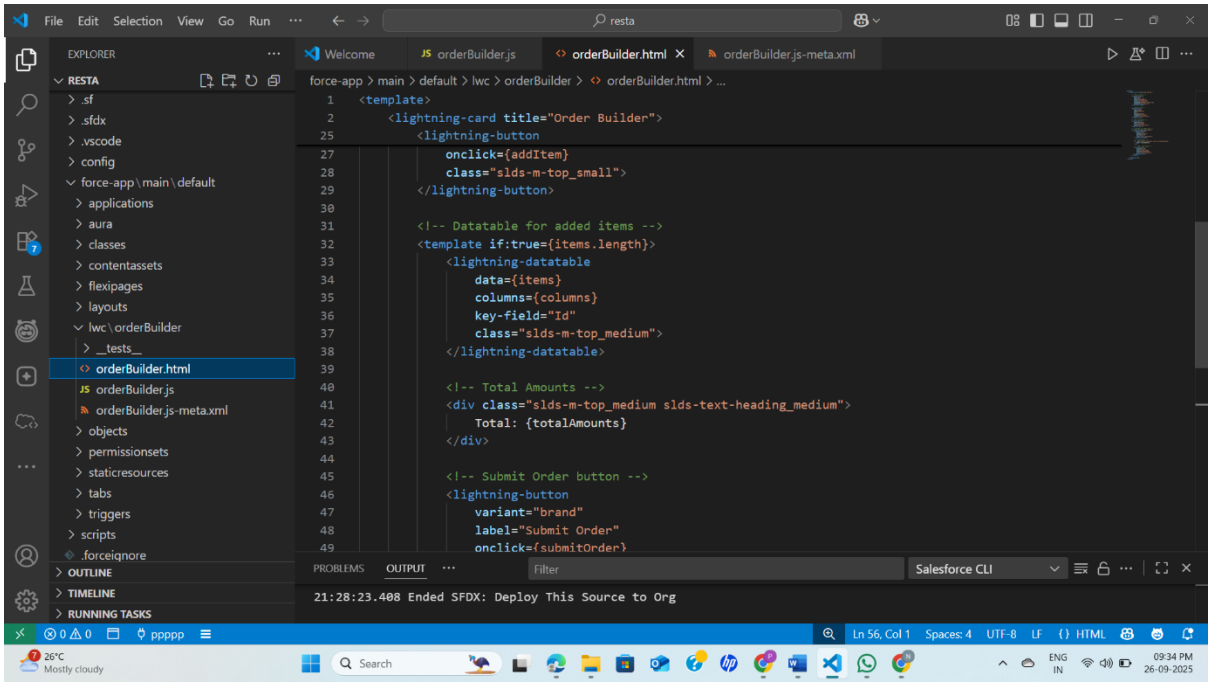
Page Details (Right Panel)

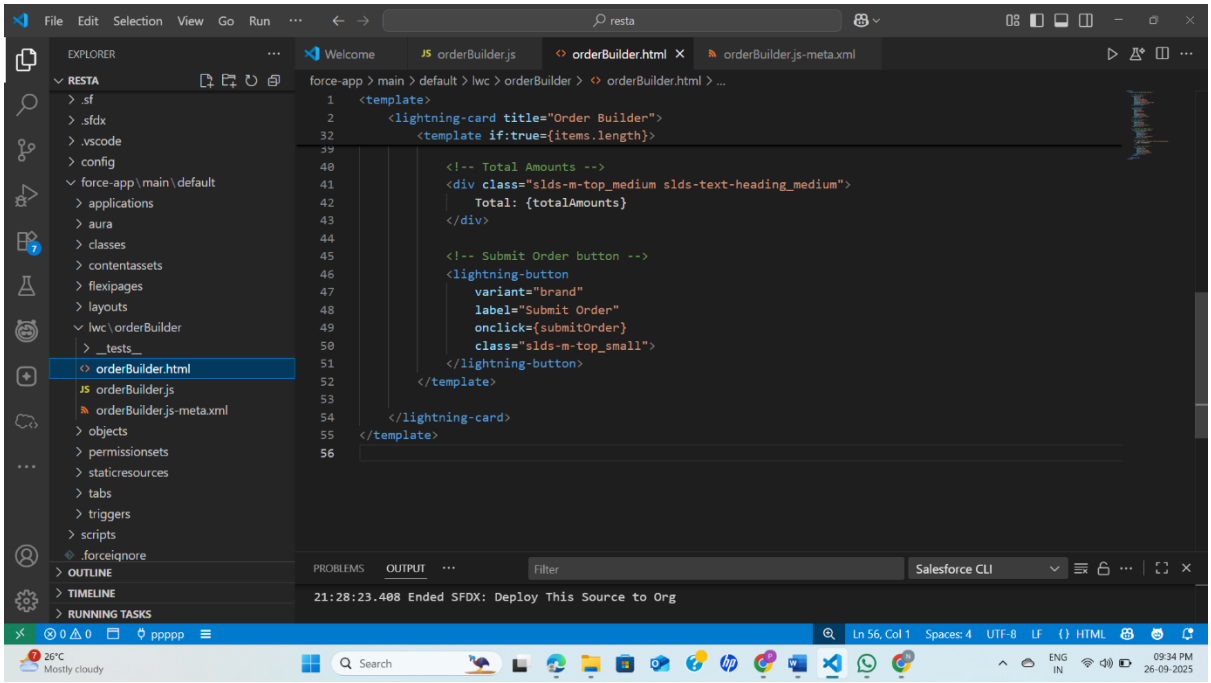
- **Label**:
  The display name of the page → Order DineIn Page.
- **API Name**:
  Internal developer name → Order_DineIn_Page.
  (This name is used in code/configuration, not shown to end-users.)
- **Page Type**:
  Record Page → means this layout is tied to a Salesforce Object's record (in your case, the **Orders** object).
  Example: When a user opens a record from the **Orders** object, this page layout will load.
- **Object**:
  Orders → This page is specifically for **Order** records.
- **Template**:
  Header and Right Sidebar → The structure of the page, defining where components can be placed.
  (Here, you see a header region, a sidebar, and the main body.)
- **Description**:
  Optional text area where you can describe the purpose of this page. Currently blank.
- **Dynamic Actions toggle**:
  Enable page-level dynamic actions for the Salesforce mobile app → lets you control actions dynamically (which buttons users see, based on conditions).



After creating the lightning app builder I have created the orderBuilder and deployed that to the org

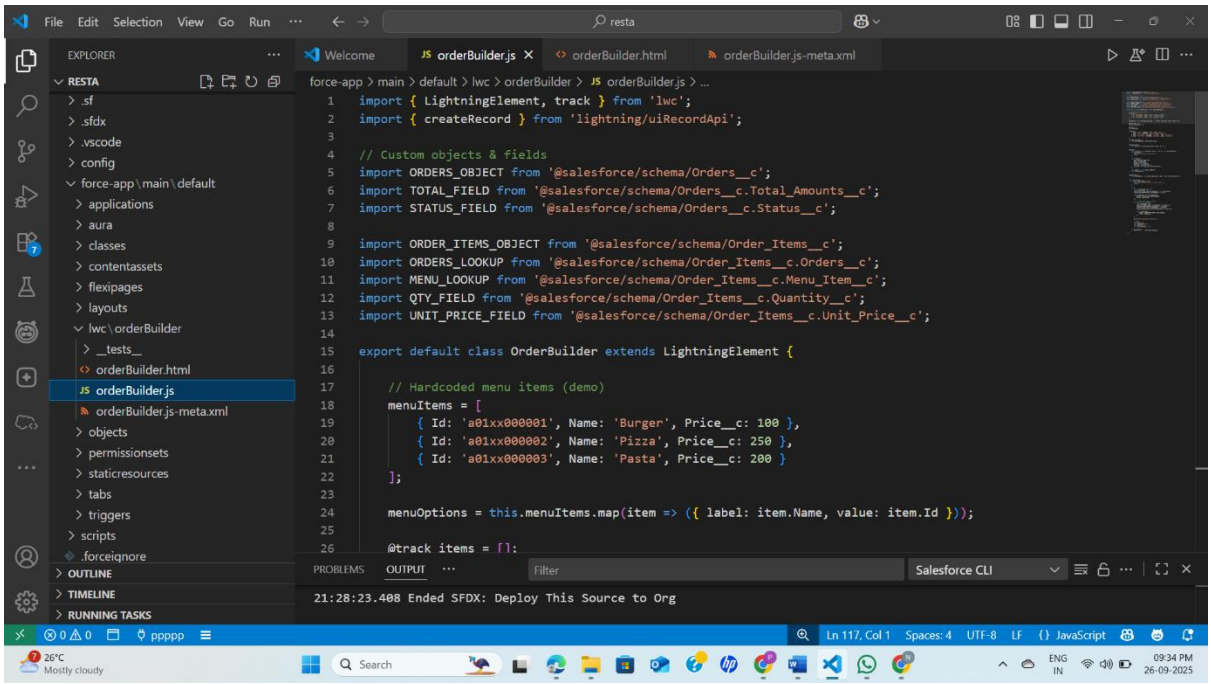


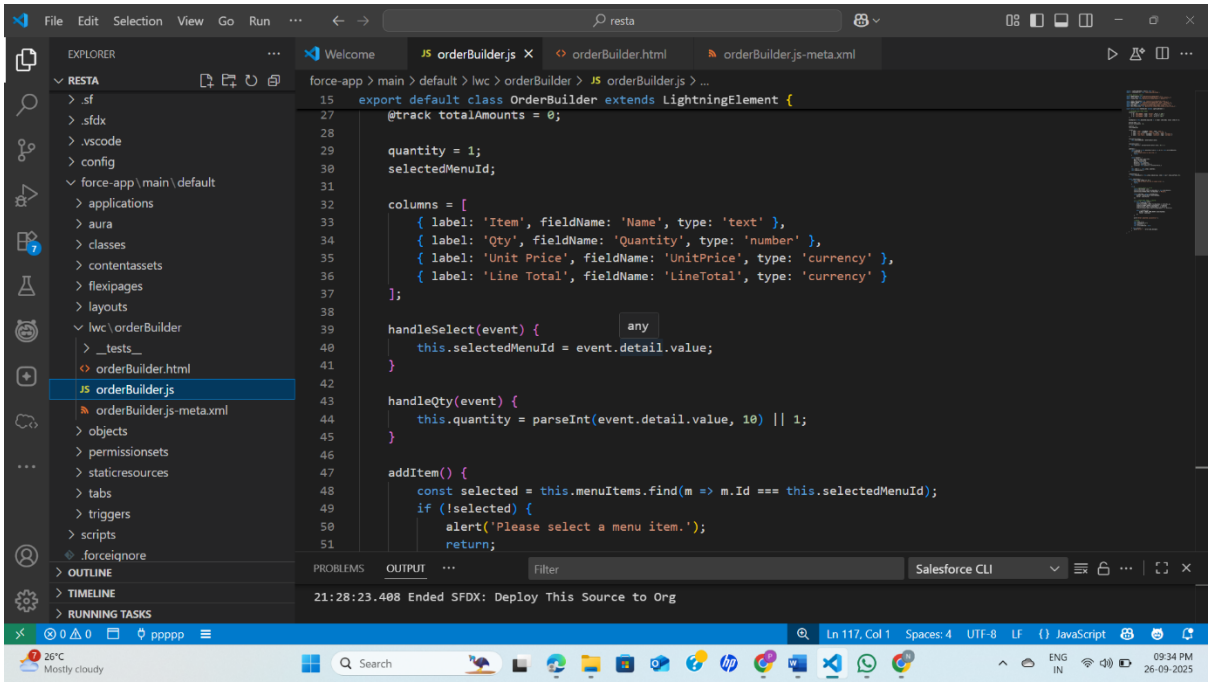


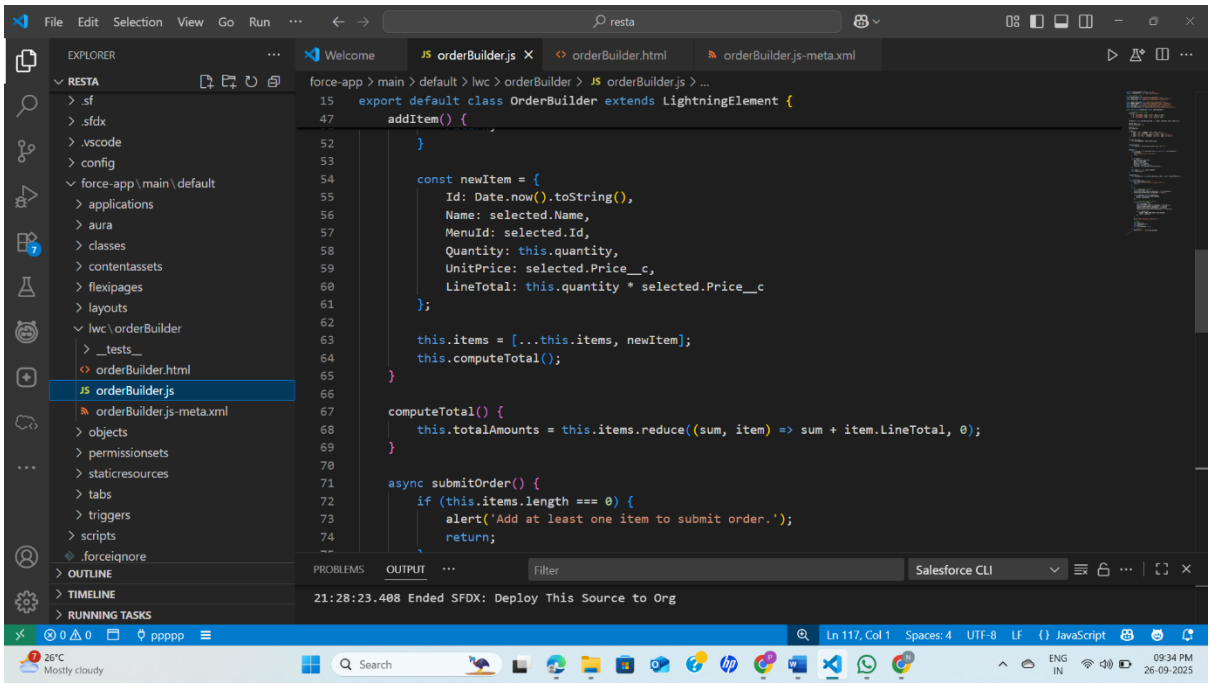

File Edit Selection View Go Run
resta
EXPLORER
RESTA
.sf
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
contentassets
flexipages
layouts
lwc\orderBuilder
__tests__
orderBuilder.html
orderBuilder.js
orderBuilder.js-meta.xml
objects
permissionsets
staticresources
tabs
triggers
scripts
.forceignore
OUTLINE
TIMELINE
RUNNING TASKS
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.html > ...
<template>
<lightning-card title="Order Builder">
<template if:true={items.length}>
<!-- Total Amounts -->

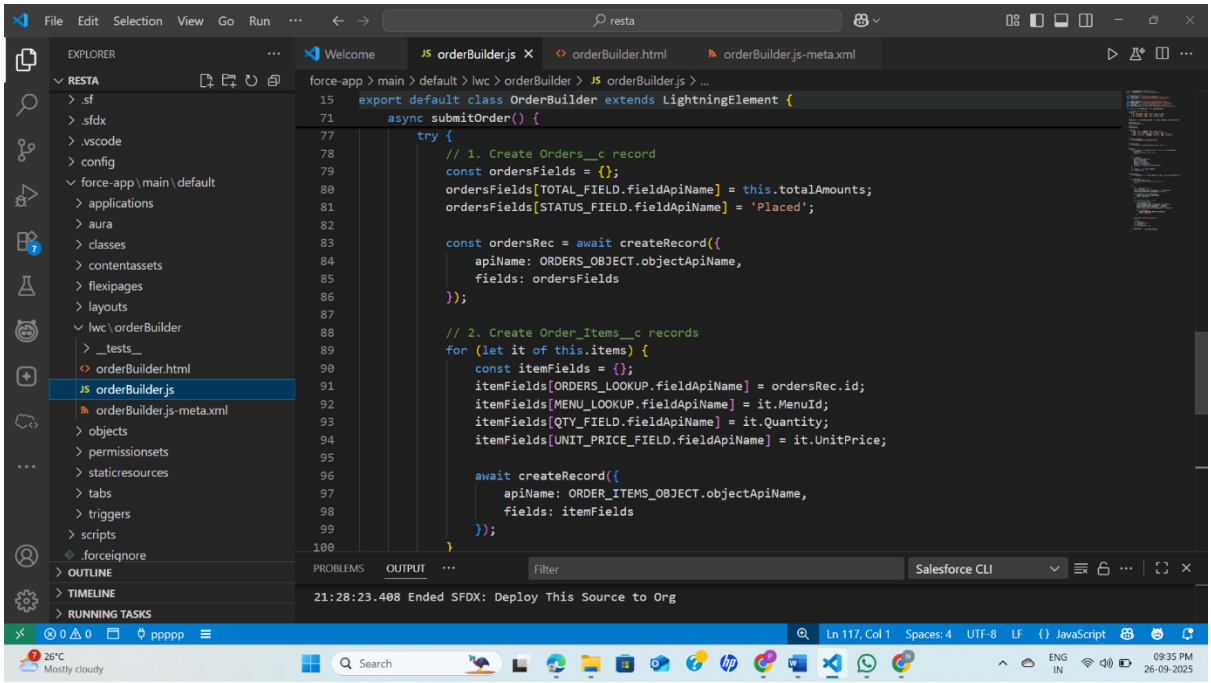
Total: {totalAmounts}

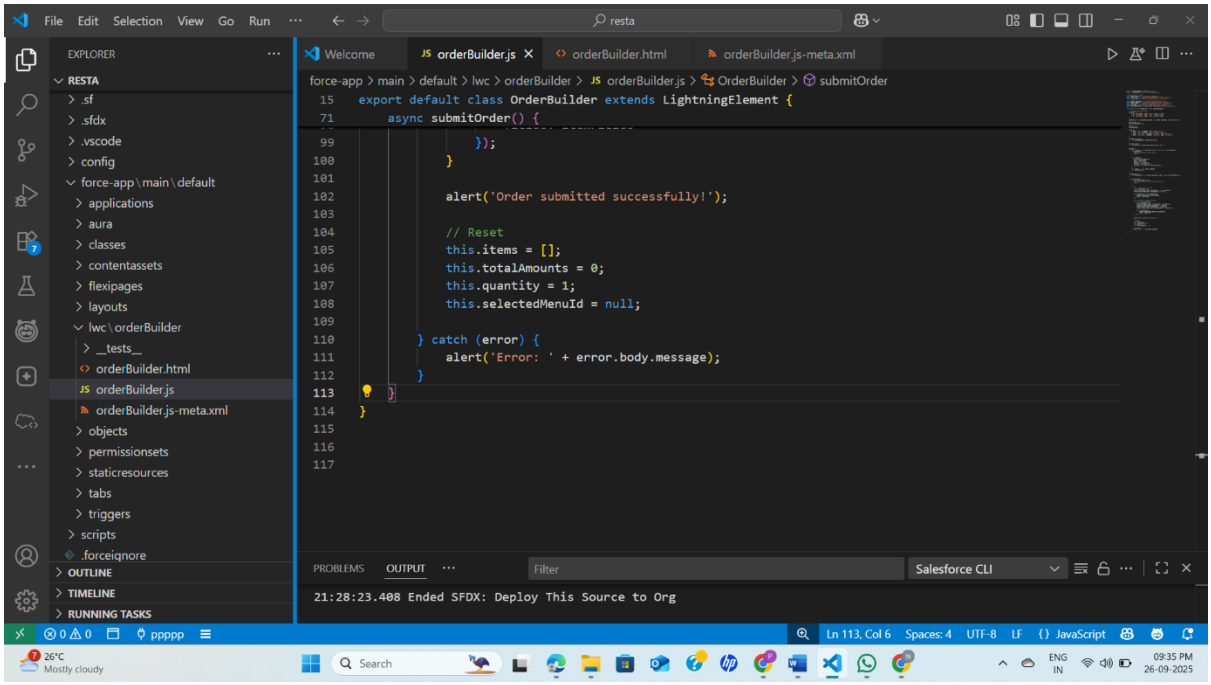
<!-- Submit Order button -->
<lightning-button
variant="brand"
label="Submit Order"
onclick={submitOrder}
class="slds-m-top_small">
</lightning-button>
</template>
</lightning-card>
</template>
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 56, Col 1
Spaces: 4
UTF-8
LF
HTML
26°C
Mostly cloudy
Search
ENG
IN
09:34 PM
26-09-2025

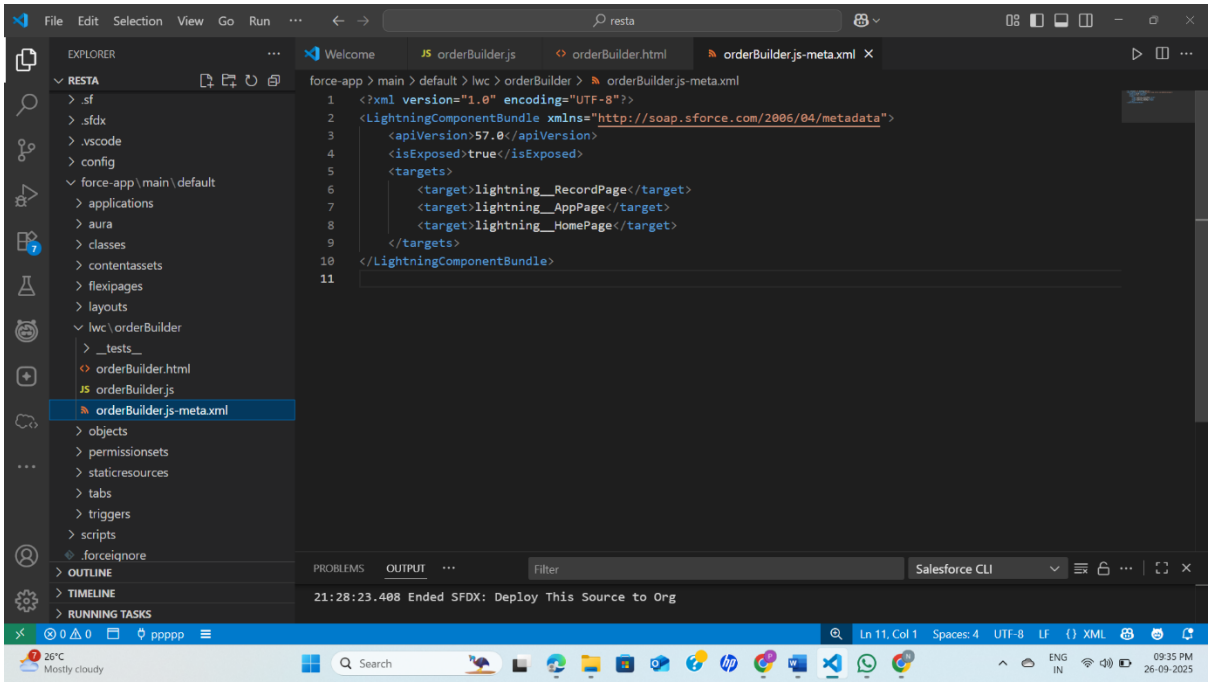

File Edit Selection View Go Run
resta
EXPLORER
RESTA
.sf
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
contentassets
flexipages
layouts
lwc\orderBuilder
__tests__
orderBuilder.html
orderBuilder.js
orderBuilder.js-meta.xml
objects
permissionsets
staticresources
tabs
triggers
scripts
.forceignore
OUTLINE
TIMELINE
RUNNING TASKS
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.js > ...
import { LightningElement, track } from 'lwc';
import { createRecord } from 'lightning/uiRecordApi';
// Custom objects & fields
import ORDERS_OBJECT from '@salesforce/schema/Orders__c';
import TOTAL_FIELD from '@salesforce/schema/Orders__c.Total_Amounts__c';
import STATUS_FIELD from '@salesforce/schema/Orders__c.Status__c';
import ORDER_ITEMS_OBJECT from '@salesforce/schema/Order_Items__c';
import ORDERS_LOOKUP from '@salesforce/schema/Order_Items__c.Orders__c';
import MENU_LOOKUP from '@salesforce/schema/Order_Items__c.Menu_Item__c';
import QTY_FIELD from '@salesforce/schema/Order_Items__c.Quantity__c';
import UNIT_PRICE_FIELD from '@salesforce/schema/Order_Items__c.Unit_Price__c';
export default class OrderBuilder extends LightningElement {
// Hardcoded menu items (demo)
menuItems = [
{ Id: 'a01xx000001', Name: 'Burger', Price__c: 100 },
{ Id: 'a01xx000002', Name: 'Pizza', Price__c: 250 },
{ Id: 'a01xx000003', Name: 'Pasta', Price__c: 200 }
];
menuOptions = this.menuItems.map(item => ({ label: item.Name, value: item.Id }));
@track items = [];
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 117, Col 1
Spaces: 4
UTF-8
LF
JavaScript
26°C
Mostly cloudy
Search
ENG
IN
09:34 PM
26-09-2025

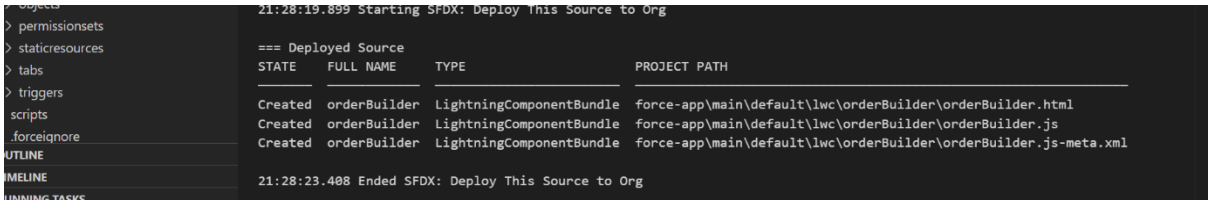

File Edit Selection View Go Run
resta
EXPLORER
RESTA
.sf
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
contentassets
flexipages
layouts
lwc\orderBuilder
__tests__
orderBuilder.html
orderBuilder.js
orderBuilder.js-meta.xml
objects
permissionsets
staticresources
tabs
triggers
scripts
.forceignore
OUTLINE
TIMELINE
RUNNING TASKS
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.js > ...
15 export default class OrderBuilder extends LightningElement {
27 @track totalAmounts = 0;
29 quantity = 1;
30 selectedMenuId;
32 columns = [
33 { label: 'Item', fieldName: 'Name', type: 'text' },
34 { label: 'Qty', fieldName: 'Quantity', type: 'number' },
35 { label: 'Unit Price', fieldName: 'UnitPrice', type: 'currency' },
36 { label: 'Line Total', fieldName: 'LineTotal', type: 'currency' }
37];
39 handleSelect(event) {
any
40 this.selectedMenuId = event.detail.value;
41 }
43 handleQty(event) {
44 this.quantity = parseInt(event.detail.value, 10) || 1;
45 }
47 addItem() {
48 const selected = this.menuItems.find(m => m.Id === this.selectedMenuId);
49 if (!selected) {
50 alert('Please select a menu item.');
51 return;
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 117, Col 1
Spaces: 4
UTF-8
LF
JavaScript
pppp
26°C
Mostly cloudy
Search
ENG
IN
09:34 PM
26-09-2025


File Edit Selection View Go Run
resta
EXPLORER
RESTA
.sf
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
contentassets
flexipages
layouts
lwc\orderBuilder
__tests__
orderBuilder.html
orderBuilder.js
orderBuilder.js-meta.xml
objects
permissionsets
staticresources
tabs
triggers
scripts
.forceignore
OUTLINE
TIMELINE
RUNNING TASKS
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.js > ...
15 export default class OrderBuilder extends LightningElement {
47 addItem() {
52 }
54 const newItem = {
55 Id: Date.now().toString(),
56 Name: selected.Name,
57 MenuId: selected.Id,
58 Quantity: this.quantity,
59 UnitPrice: selected.Price__c,
60 LineTotal: this.quantity * selected.Price__c
61 };
63 this.items = [...this.items, newItem];
64 this.computeTotal();
65 }
67 computeTotal() {
68 this.totalAmounts = this.items.reduce((sum, item) => sum + item.LineTotal, 0);
69 }
71 async submitOrder() {
72 if (this.items.length === 0) {
73 alert('Add at least one item to submit order.');
74 return;
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 117, Col 1
Spaces: 4
UTF-8
LF
JavaScript
pppp
26°C
Mostly cloudy
Search
ENG
IN
09:34 PM
26-09-2025


File Edit Selection View Go Run
resta
EXPLORER
RESTA
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.js > ...
export default class OrderBuilder extends LightningElement {
async submitOrder() {
try {
// 1. Create Orders__c record
const ordersFields = {};
ordersFields[TOTAL_FIELD.fieldApiName] = this.totalAmounts;
ordersFields[STATUS_FIELD.fieldApiName] = 'Placed';
const ordersRec = await createRecord({
apiName: ORDERS_OBJECT.objectApiName,
fields: ordersFields
});
// 2. Create Order_Items__c records
for (let it of this.items) {
const itemFields = {};
itemFields[ORDERS_LOOKUP.fieldApiName] = ordersRec.id;
itemFields[MENU_LOOKUP.fieldApiName] = it.MenuId;
itemFields[QTY_FIELD.fieldApiName] = it.Quantity;
itemFields[UNIT_PRICE_FIELD.fieldApiName] = it.UnitPrice;
await createRecord({
apiName: ORDER_ITEMS_OBJECT.objectApiName,
fields: itemFields
});
}
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 117, Col 1
Spaces: 4
UTF-8
LF
JavaScript
26°C
Mostly cloudy
Search
ENG
IN
09:35 PM
26-09-2025


File Edit Selection View Go Run
resta
EXPLORER
RESTA
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.js > OrderBuilder > submitOrder
export default class OrderBuilder extends LightningElement {
async submitOrder() {
});
}
alert('Order submitted successfully!');
// Reset
this.items = [];
this.totalAmounts = 0;
this.quantity = 1;
this.selectedMenuId = null;
} catch (error) {
alert('Error: ' + error.body.message);
}
}
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 113, Col 6
Spaces: 4
UTF-8
LF
JavaScript
26°C
Mostly cloudy
Search
ENG
IN
09:35 PM
26-09-2025

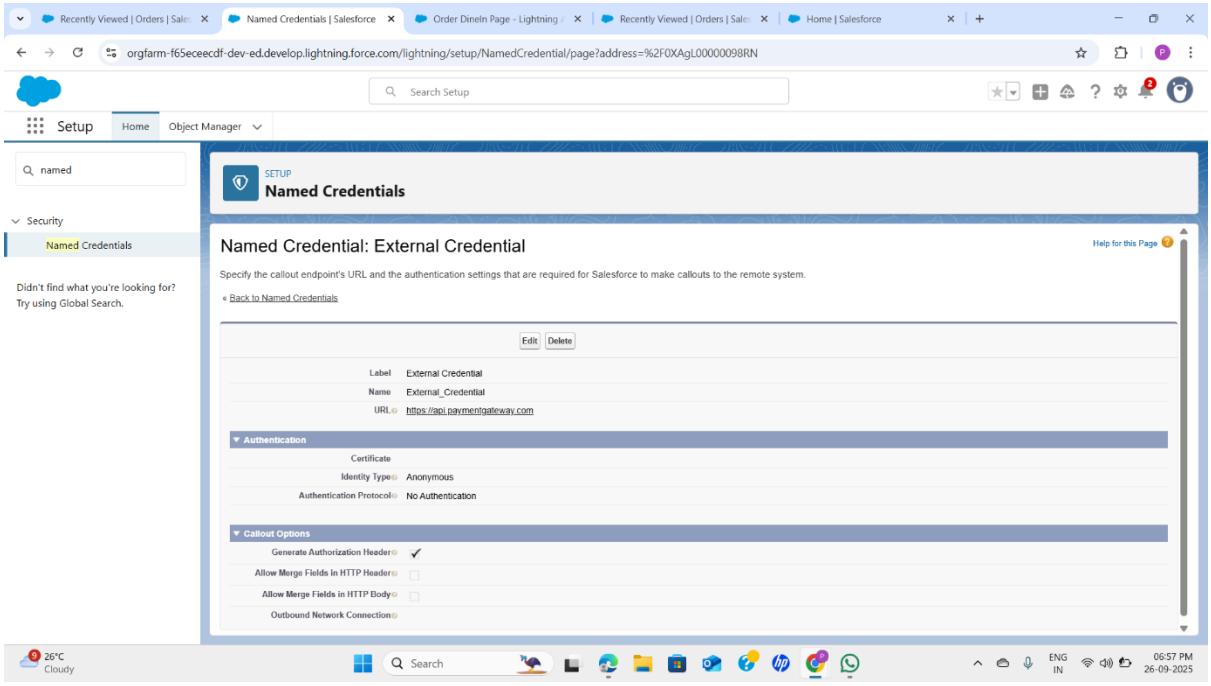

File Edit Selection View Go Run
resta
EXPLORER
RESTA
.sf
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
contentassets
flexipages
layouts
lwc\orderBuilder
__tests__
orderBuilder.html
orderBuilder.js
orderBuilder.js-meta.xml
objects
permissionsets
staticresources
tabs
triggers
scripts
.forceignore
OUTLINE
TIMELINE
RUNNING TASKS
Welcome
orderBuilder.js
orderBuilder.html
orderBuilder.js-meta.xml
force-app > main > default > lwc > orderBuilder > orderBuilder.js-meta.xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>57.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__RecordPage</target>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
</targets>
</LightningComponentBundle>
PROBLEMS
OUTPUT
Filter
Salesforce CLI
21:28:23.408 Ended SFDX: Deploy This Source to Org
Ln 11, Col 1
Spaces: 4
UTF-8
LF
XML
26°C
Mostly cloudy
Search
ENG
IN
09:35 PM
26-09-2025


permissionsets
staticresources
tabs
triggers
scripts
.forceignore
21:28:19.899 Starting SFDX: Deploy This Source to Org
=== Deployed Source
STATE FULL NAME TYPE PROJECT PATH
Created orderBuilder LightningComponentBundle force-app\main\default\lwc\orderBuilder\orderBuilder.html
Created orderBuilder LightningComponentBundle force-app\main\default\lwc\orderBuilder\orderBuilder.js
Created orderBuilder LightningComponentBundle force-app\main\default\lwc\orderBuilder\orderBuilder.js-meta.xml
21:28:23.408 Ended SFDX: Deploy This Source to Org

PHASE 7

Named Credentials

1. **Label & Name**
   - Label: PaymentGateway → Human-readable name
   - Name: PaymentGateway → API reference for Apex callouts
2. **URL**
   - External system endpoint URL, e.g., https://api.paymentgateway.com
   - This will be the base URL for all callouts using this Named Credential
3. **Identity Type**
   - Choose **Named Principal** → Single set of credentials used by all users
   - Optionally, you could choose **Per User** if each Salesforce user needs separate authentication
4. **Authentication**
   - If you have created an **External Credential** (OAuth, AWS SigV4), select it here
   - Otherwise, choose **Password Authentication** and enter username/password or token
5. **Generate Authorization Header** (optional)
   - For OAuth or token-based auth, Salesforce automatically adds the Authorization header to requests



Use Platform Events for Real-Time Notifications

Notify external systems when orders are updated

1. **Create a Platform Event**
    - Setup → Platform Events → New Platform Event
    - Name: OrderStatusEvent
    - Fields:
        - OrderId (Text)
        - Status (Picklist: Pending, Completed, Cancelled)
        - CustomerId (Text)
2. Platform Events are a **Salesforce event-driven messaging mechanism**.
3. They allow **Salesforce to notify external systems** (POS, delivery platforms, analytics tools) **in real-time** when something changes in Salesforce.
4. Think of it as **Salesforce "publishing" an event** that subscribers can react to immediately.

 Real-Time Communication

- External systems can subscribe via **CometD protocol** or **Streaming API**.
- Notifications are near-instant, no need for polling Salesforce.

 Custom Event Structure

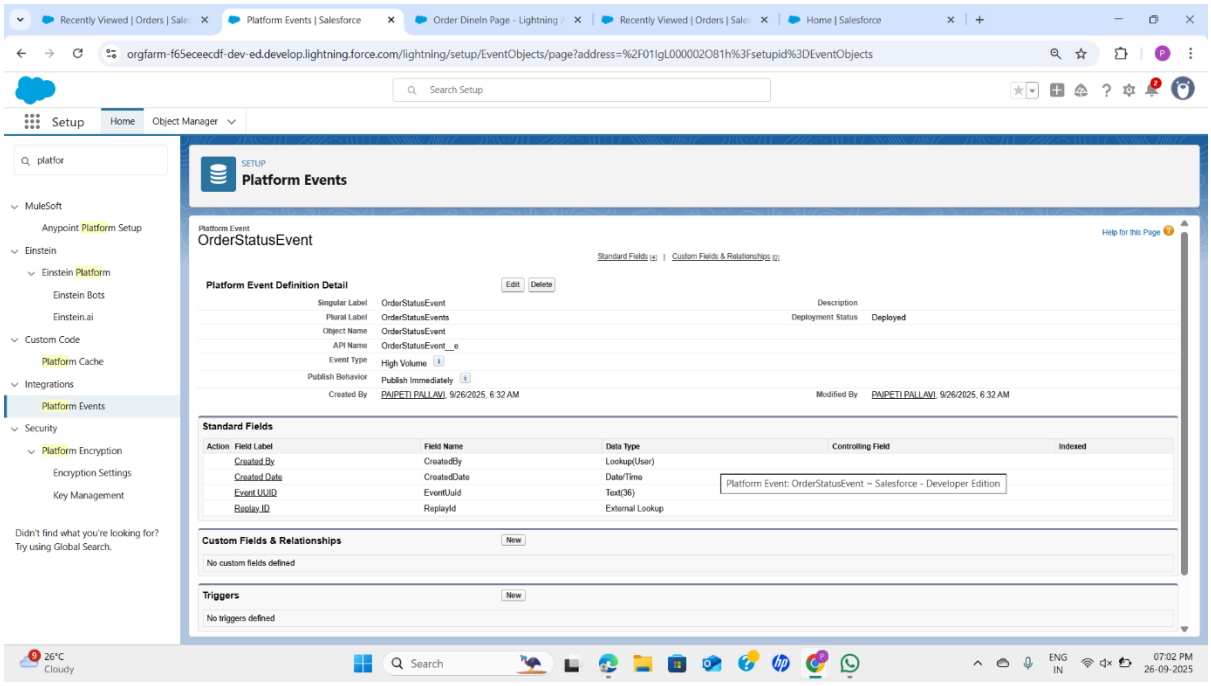
- You define which fields are sent in the event.
- Only necessary data is included, making it lightweight.

 Reliable Delivery

- Events can be replayed if an external system was temporarily disconnected.

 Supports Standard & Custom Subscribers

- Subscribers can be **Apex triggers**, **Lightning components**, **external middleware**, or other apps.



Change Data Capture

Automatically streams record changes for standard/custom objects

1. Setup → Change Data Capture → Select Object (e.g., Order__c)
2. Enable CDC → External system can subscribe for **insert, update, delete** events.

1. □ **Change Data Capture (CDC)** is a Salesforce **event-driven mechanism** that automatically **streams changes of Salesforce records** to external systems in real-time.
2. □ It is **especially useful for integrations**, so external systems (like POS, ERP, or analytics platforms) **stay in sync** with Salesforce without constant polling.

□ Real-Time Streaming

- Sends events immediately when a record is **created, updated, deleted, or undeleted**.
- Reduces the need for batch jobs or polling APIs.

□ Works with Standard & Custom Objects

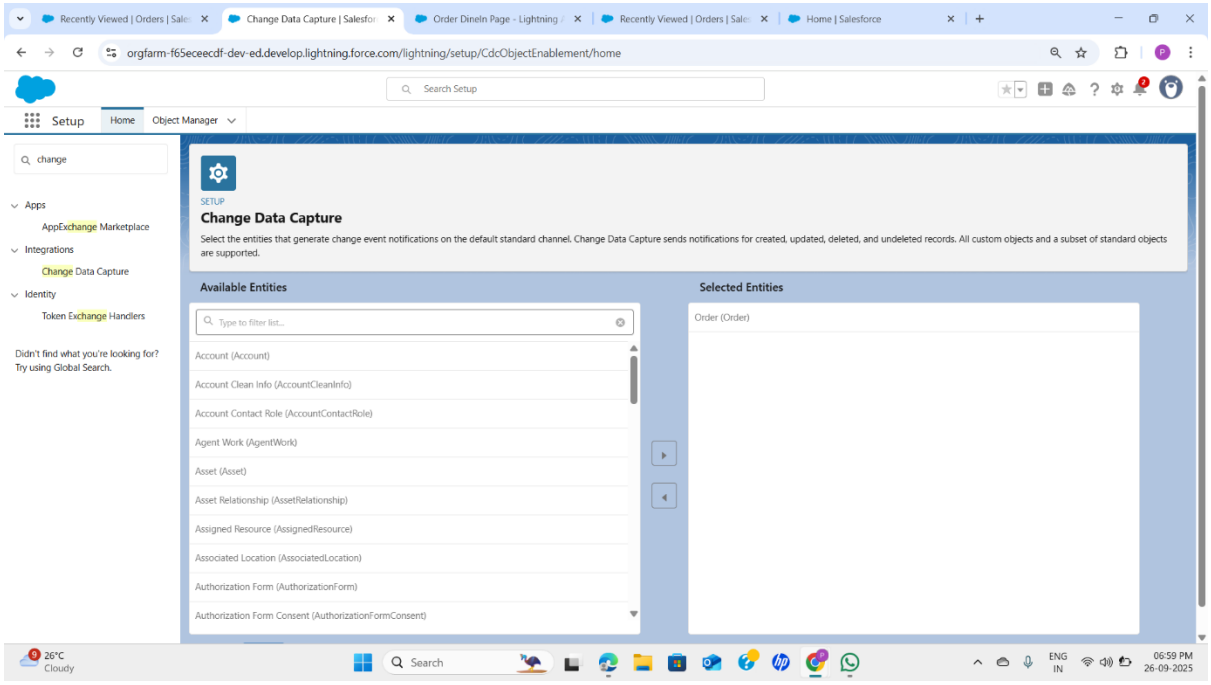
- Example objects:
  - Standard: Account, Contact, Order
  - Custom: Order__c, Booking__c

□ Reliable & Secure

- Events are delivered via **CometD protocol** or **Streaming API**.
- Can be **subscribed to by external systems** using secure authentication.

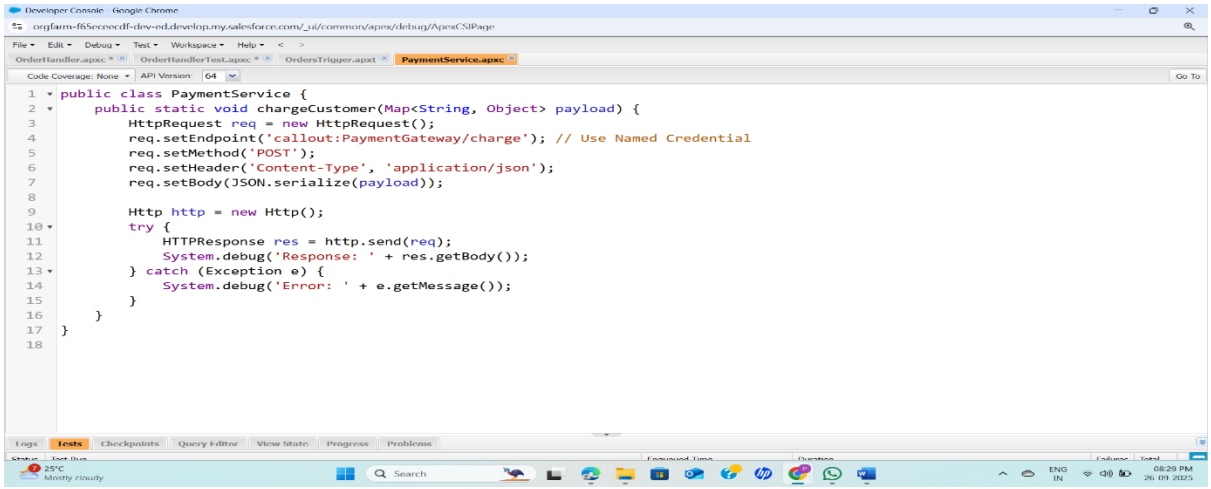
✅ Supports Bulk Changes

- Multiple changes can be sent in a single event batch, optimizing API usage.



Apex Callouts Using Named Credentials

- An **Apex callout** is when Salesforce (using Apex code) makes an **HTTP request** to an external system's API (e.g., Payment Gateway, Twilio, POS).
- Used for:
  - Sending data (POST) → e.g., payment requests
  - Fetching data (GET) → e.g., order status from POS
  - Updating data (PATCH/PUT)
  - Deleting data (DELETE)



PHASE 8

Data Export

 Go to Setup

- Click the **gear icon** (⚙) in the top-right corner of Salesforce.
- Click **Setup**.

 Search for Data Export

- In the Quick Find box (top-left), type: Data Export.
- Click **Data Export** under *Data Management*.

 Choose Export Type

- **Export Now** → one-time backup.
- **Schedule Export** → automatically back up weekly/monthly.
    - Only available in certain Salesforce editions.

 Select Objects to Backup

- Check **"Select All"** to include all standard and custom objects.
- **Include attachments, documents, Salesforce Files** if needed.

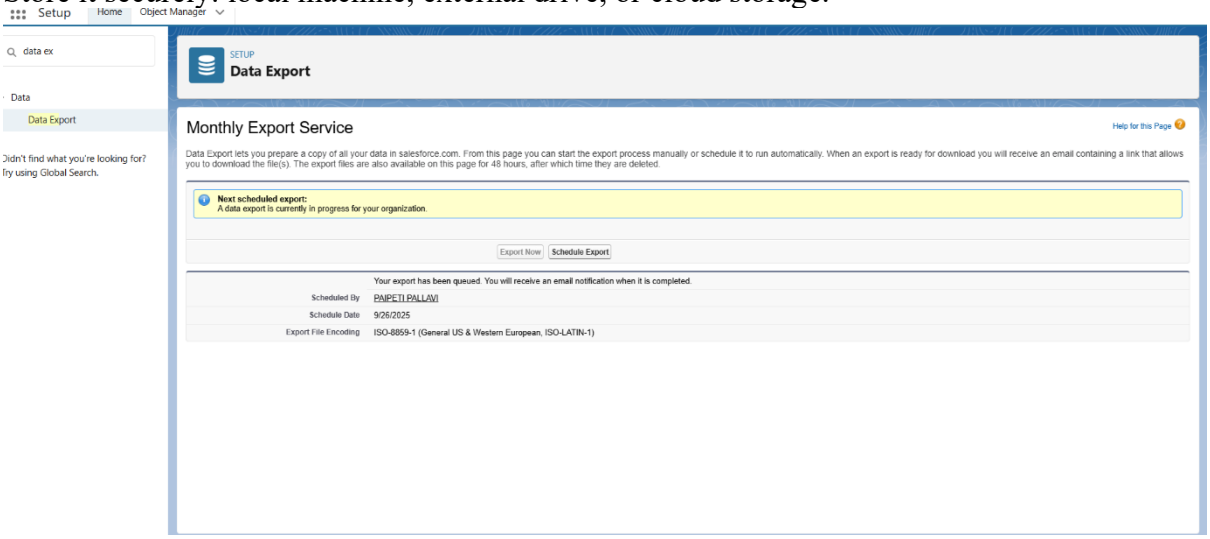
 Start the Export

- Click **Start Export**.
- Salesforce will prepare a ZIP file with your data.
- **Note:** This may take hours depending on how much data you have.

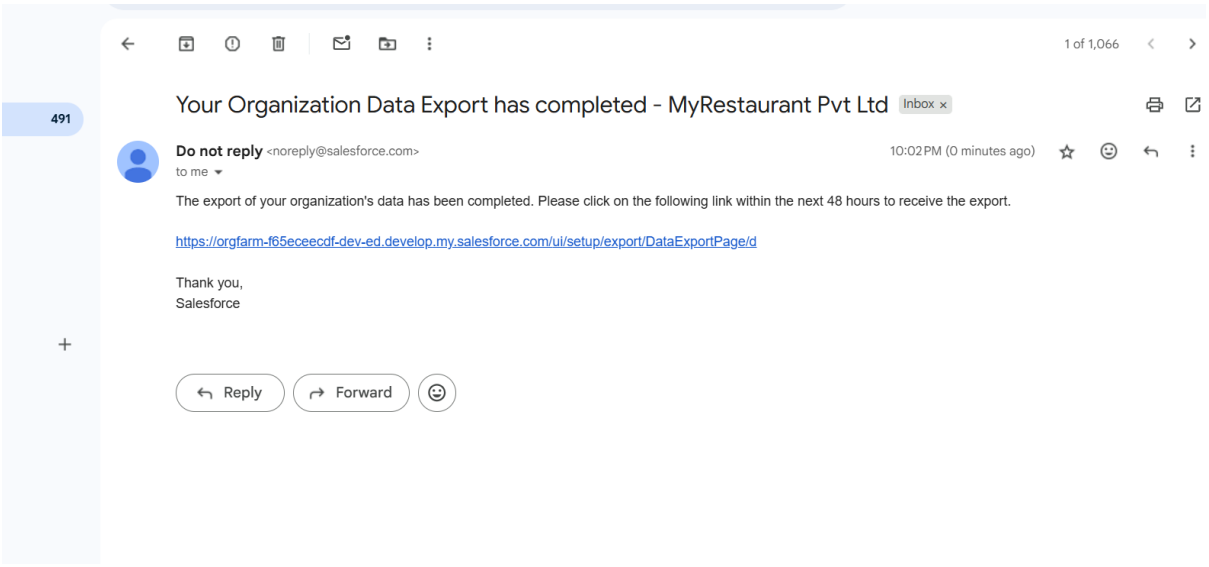
 Download & Store Safely

- You will get an email when the export is ready.
- Go to the link, download the ZIP file.

- Store it securely: local machine, external drive, or cloud storage.



- After this export I got the email like this.



Data Import

Go to Setup → Data Import Wizard

- Click the gear icon → Setup → Quick Find: Data Import Wizard → Launch Wizard.

Choose Object

- Select the object you want to import (e.g., Contacts, Accounts, Orders).

Select Operation

- **Add New Records** → only insert.
- **Update Existing Records** → update by matching Salesforce ID or External ID.
- **Upsert** → create or update using External ID (recommended if updating historical data).

Upload CSV File

- Prepare CSV: include required fields and External ID fields if needed.
- Click **Choose CSV file** and upload it.

Map Fields

- Salesforce will attempt to auto-map fields from your CSV.
- Verify all important fields (Name, Email, Order Number, Dates, etc.) are mapped correctly.

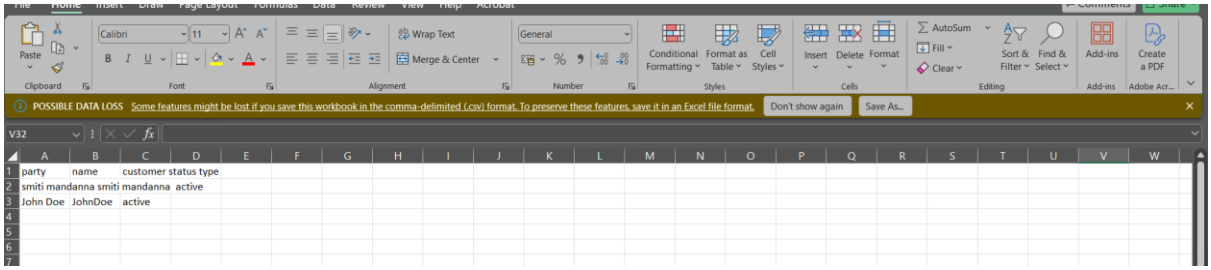
Run Import

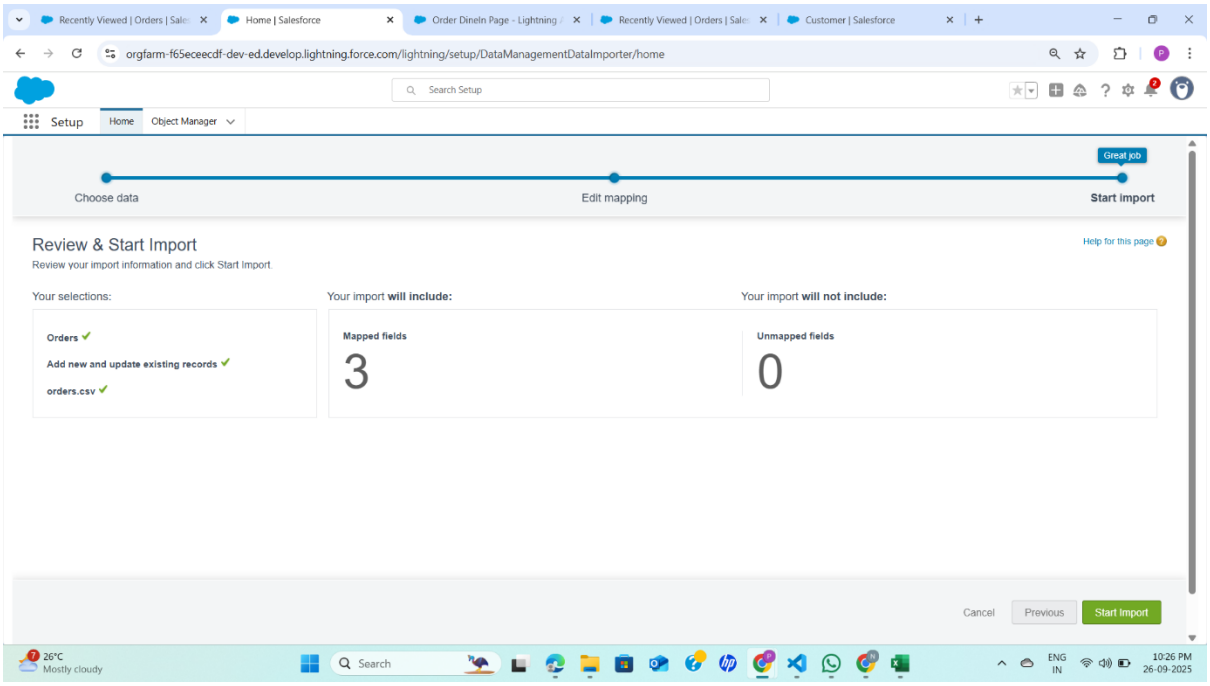
- Click **Start Import**.
- Wait for it to finish; you can **monitor status** in Data Import Wizard.

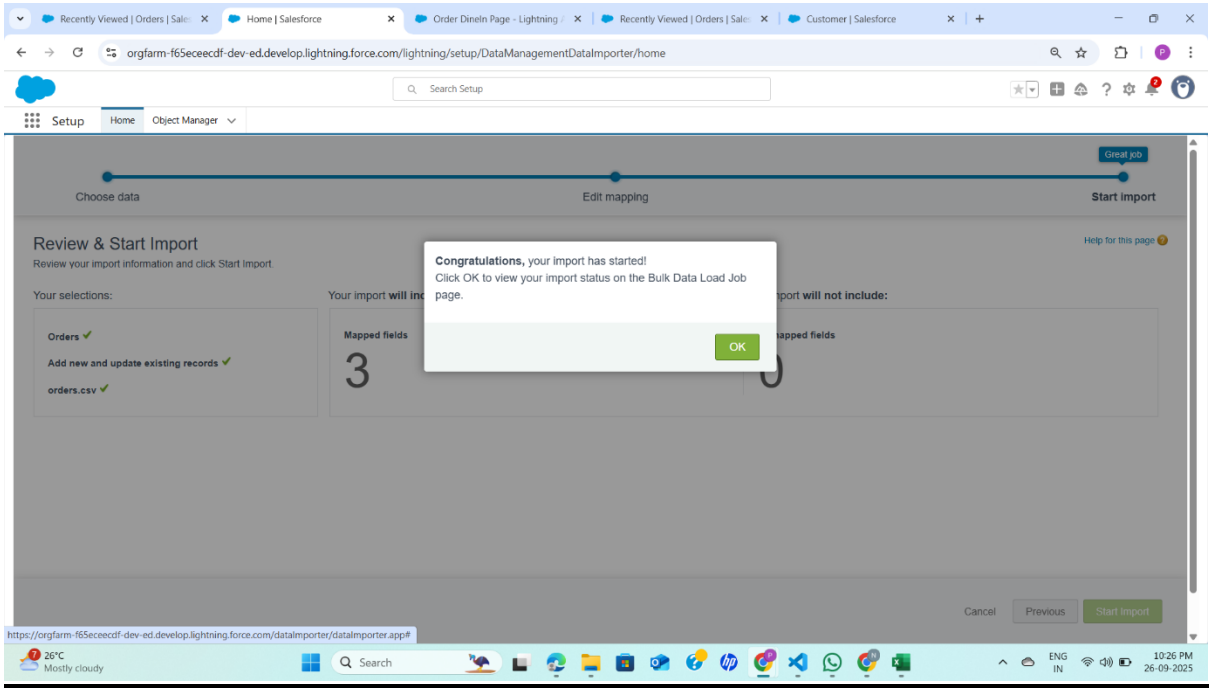
Check Results

- Download **Success** and **Error** CSVs.
- Fix any errors (required fields missing, duplicate rule conflicts, invalid picklist values) and re-import if needed.

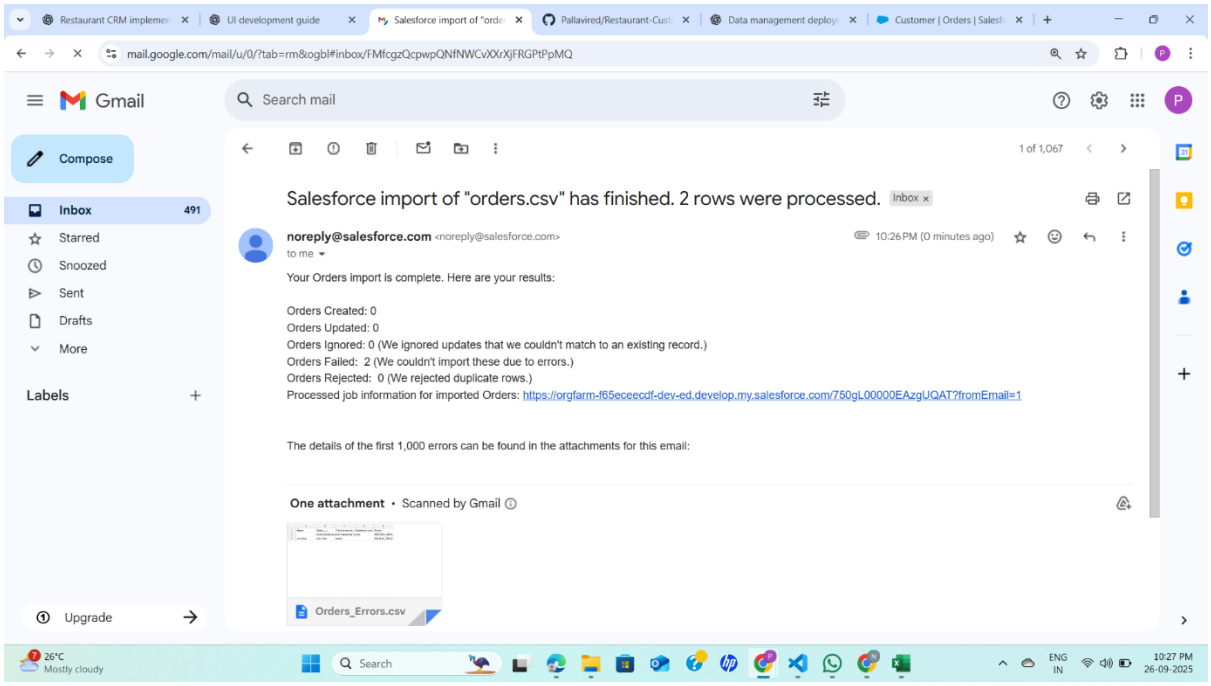
I have created the csv file for the orders







After importing I have received the email like this


Gmail
Search mail
Compose
Inbox 491
Starred
Snoozed
Sent
Drafts
More
Labels
Upgrade
1 of 1,067
Salesforce import of "orders.csv" has finished. 2 rows were processed.
Inbox
noreply@salesforce.com <noreply@salesforce.com>
10:26 PM (0 minutes ago)
to me
Your Orders import is complete. Here are your results:
Orders Created: 0
Orders Updated: 0
Orders Ignored: 0 (We ignored updates that we couldn't match to an existing record.)
Orders Failed: 2 (We couldn't import these due to errors.)
Orders Rejected: 0 (We rejected duplicate rows.)
Processed job information for imported Orders: https://orgfarm-f65eceecdf-dev-ed.develop.my.salesforce.com/750gL00000EAzgUQAT?fromEmail=1
The details of the first 1,000 errors can be found in the attachments for this email:
One attachment • Scanned by Gmail
Orders_Errors.csv
26°C
Mostly cloudy
Search
ENG
IN
10:27 PM
26-09-2025

PHASE 9

REPORTS AND DASHBOARDS:

What is a Report in Salesforce?

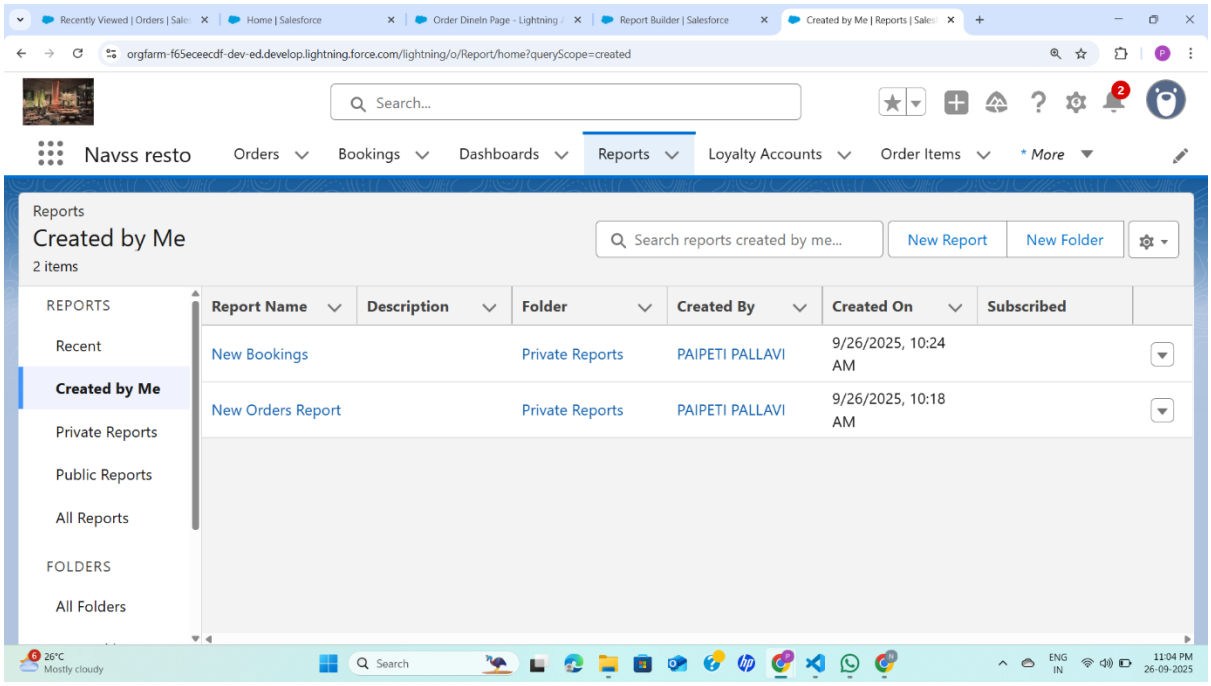
A **Report** is a list of records that meet certain criteria, displayed in rows and columns.
It helps you **analyze data**, **track performance**, and **make decisions**.

For example:

- A report on **Orders** could show: *Order Number, Account, Order Amount, Status*.
- A report on **Customers** could show: *Customer Name, Total Purchases, Loyalty Points*.

🧩 Key Elements of a Report

1. **Report Type**
   - Defines *which object(s)* you're reporting on (e.g., Orders, Accounts, Contacts, or Orders with Order Items).
   - Determines what fields are available to add in your report.
2. **Columns**
   - The fields you choose to display (e.g., Order Number, Account Name, Order Start Date).
3. **Filters**
   - Narrow down the data (e.g., *Status = Completed*, *Created Date = Last 30 Days*).
4. **Grouping**
   - Organize records by field values (e.g., group Orders by *Status* or *Account*).
5. **Summaries**
   - Totals, averages, minimum, maximum values on numeric fields (e.g., total Order Amount).
6. **Report Formats** (in Lightning)
   - **Tabular** → Simple list (like an Excel table).
   - **Summary** → Grouped rows, with subtotals.
   - **Matrix** → Group rows *and* columns (like a pivot table).
   - **Joined** → Multiple report blocks combined for comparison.



Dashboards

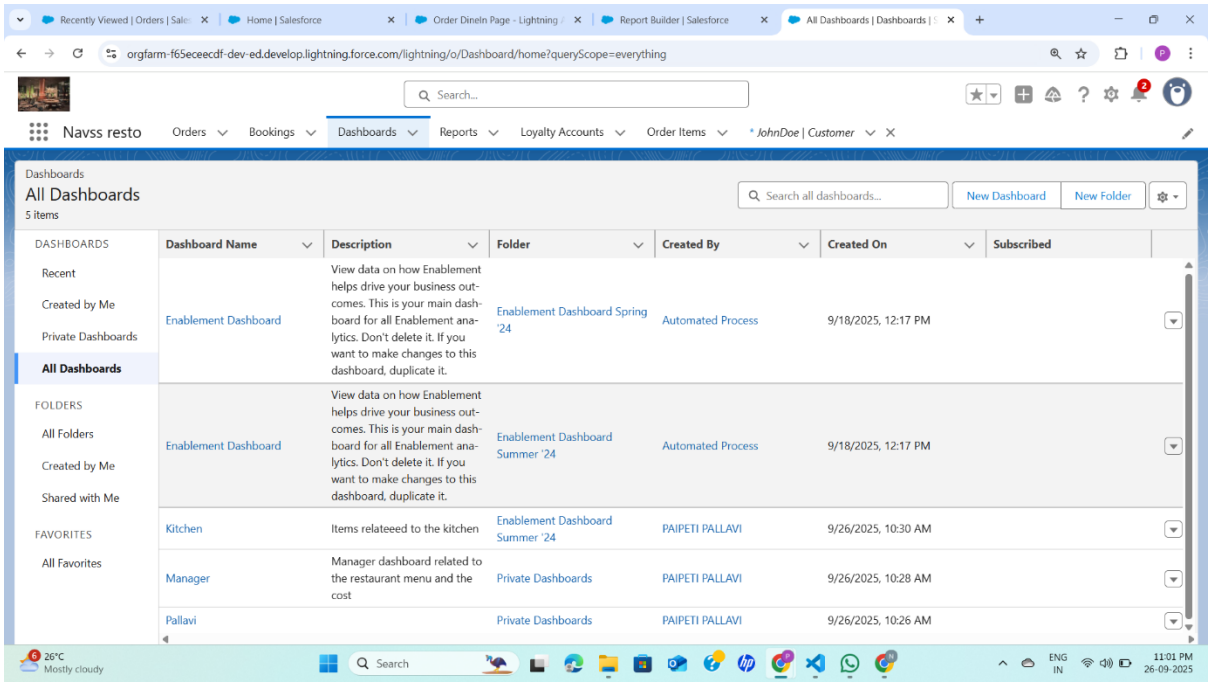
- **Purpose:**

This dashboard provides insights into how enablement activities (such as training, guidance, or support programs) drive business outcomes. It is designed to be the central place for monitoring enablement analytics.

- **Features:**

  - Tracks business impact of enablement strategies.
  - Shows KPIs aligned to business performance.
  - Helps leadership and managers evaluate enablement ROI.

- **Notes:** This dashboard should not be deleted. If changes are required, it must be duplicated before editing.


Recently Viewed | Orders | Sale
Home | Salesforce
Order DineIn Page - Lightning
Report Builder | Salesforce
All Dashboards | Dashboards |
orgfarm-f65eceecdf-dev-ed.develop.lightning.force.com/lightning/o/Dashboard/home?queryScope=everything
Search...
Navss resto
Orders
Bookings
Dashboards
Reports
Loyalty Accounts
Order Items
* JohnDoe | Customer
Dashboards
All Dashboards
5 items
Search all dashboards...
New Dashboard
New Folder
DASHBOARDS
Recent
Created by Me
Private Dashboards
All Dashboards
FOLDERS
All Folders
Created by Me
Shared with Me
FAVORITES
All Favorites
Dashboard Name
Description
Folder
Created By
Created On
Subscribed
Enablement Dashboard
View data on how Enablement helps drive your business outcomes. This is your main dashboard for all Enablement analytics. Don't delete it. If you want to make changes to this dashboard, duplicate it.
Enablement Dashboard Spring '24
Automated Process
9/18/2025, 12:17 PM
Enablement Dashboard
View data on how Enablement helps drive your business outcomes. This is your main dashboard for all Enablement analytics. Don't delete it. If you want to make changes to this dashboard, duplicate it.
Enablement Dashboard Summer '24
Automated Process
9/18/2025, 12:17 PM
Kitchen
Items relateeed to the kitchen
Enablement Dashboard Summer '24
PAIPETI PALLAVI
9/26/2025, 10:30 AM
Manager
Manager dashboard related to the restaurant menu and the cost
Private Dashboards
PAIPETI PALLAVI
9/26/2025, 10:28 AM
Pallavi
Private Dashboards
PAIPETI PALLAVI
9/26/2025, 10:26 AM
26°C
Mostly cloudy
Search
ENG
IN
11:01 PM
26-09-2025