Fuel Station CRM Project

**Problem statement:**Fuel stations face challenges in efficiently managing suppliers, fuel inventory, customer transactions, loyalty programs, and reporting. Manual processes or disconnected systems often result in data errors, delayed decision-making, and missed business opportunities.  
  
The goal of this CRM project is to create a centralized Salesforce-based solution for managing all aspects of a fuel station chain: suppliers, buyers, fuel stock, gas stations, customer details, and loyalty rewards. This system should streamline operations, improve customer service, and provide real-time insights into business performance.

# Phase 1 – Problem Understanding & Industry Analysis

Objective: Establish a clear understanding of the industry challenges and define the scope of the CRM solution.

**Requirement Gathering**: A structured process involving direct discussions with administrators, suppliers, station managers, and buyers. The aim is to capture both functional and non-functional requirements, such as supplier onboarding, real-time stock management, buyer account maintenance, and loyalty program implementation. This ensures the CRM system is built around actual business needs.

**Stakeholder Analysis**: A detailed examination of all involved roles, including Admins (system governance), Suppliers (fuel providers), Gas Station Managers (daily operations), and Buyers (customers). Each role's expectations, challenges, and responsibilities are documented to ensure the system delivers value to every stakeholder.

**Business Process Mapping**: Mapping the complete workflow, starting from a fuel request raised by the station manager, moving through supplier approval and delivery, stock updates, and ending with customer transactions and reporting. This end-to-end visualization highlights inefficiencies and forms the basis for automation and system design.

**Industry-Specific Use Cases**: Identifying specialized needs of the oil and gas industry such as regulatory compliance, precise fuel supply monitoring, and supplier automation. These cases guide the customization of the CRM to handle industry challenges effectively.

**AppExchange Exploration**: Analyzing Salesforce’s AppExchange marketplace to evaluate existing Oil & Gas solutions. Although these provide useful insights, a custom solution is chosen to meet the unique operational needs of fuel stations more effectively.

✅ Outcome: Outcome: Clear understanding of project goals, requirements, and industry context to guide CRM design.

# Phase 2 – Org Setup & Configuration

Objective: Prepare the Salesforce environment for fuel station CRM implementation.

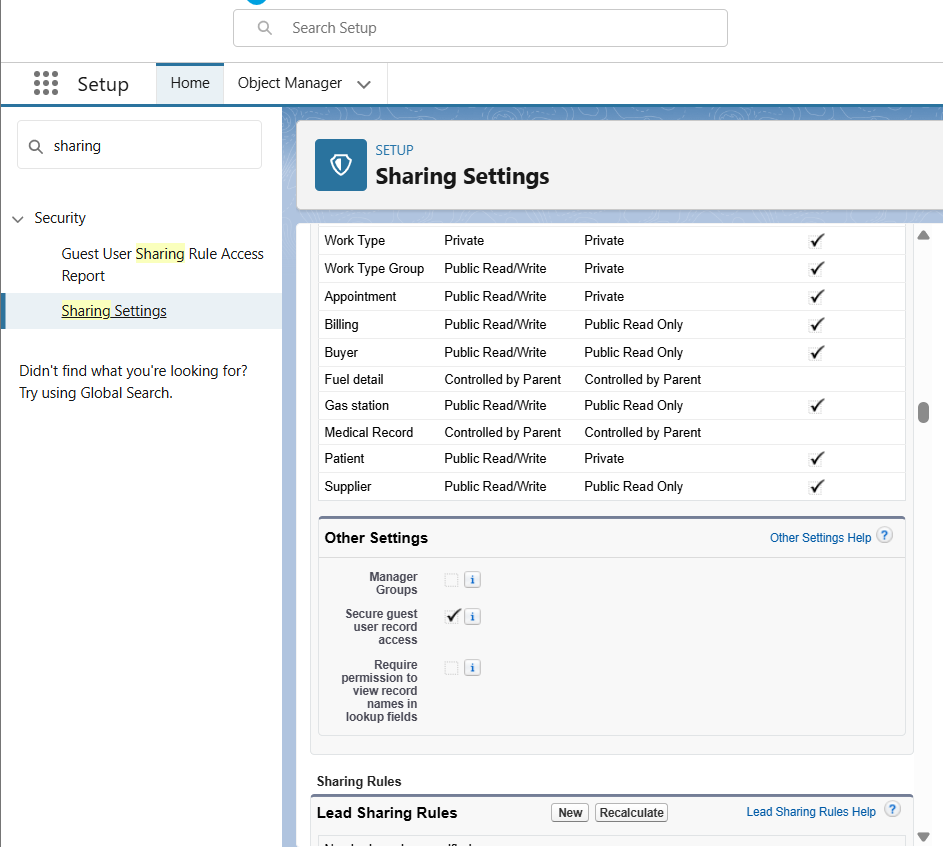
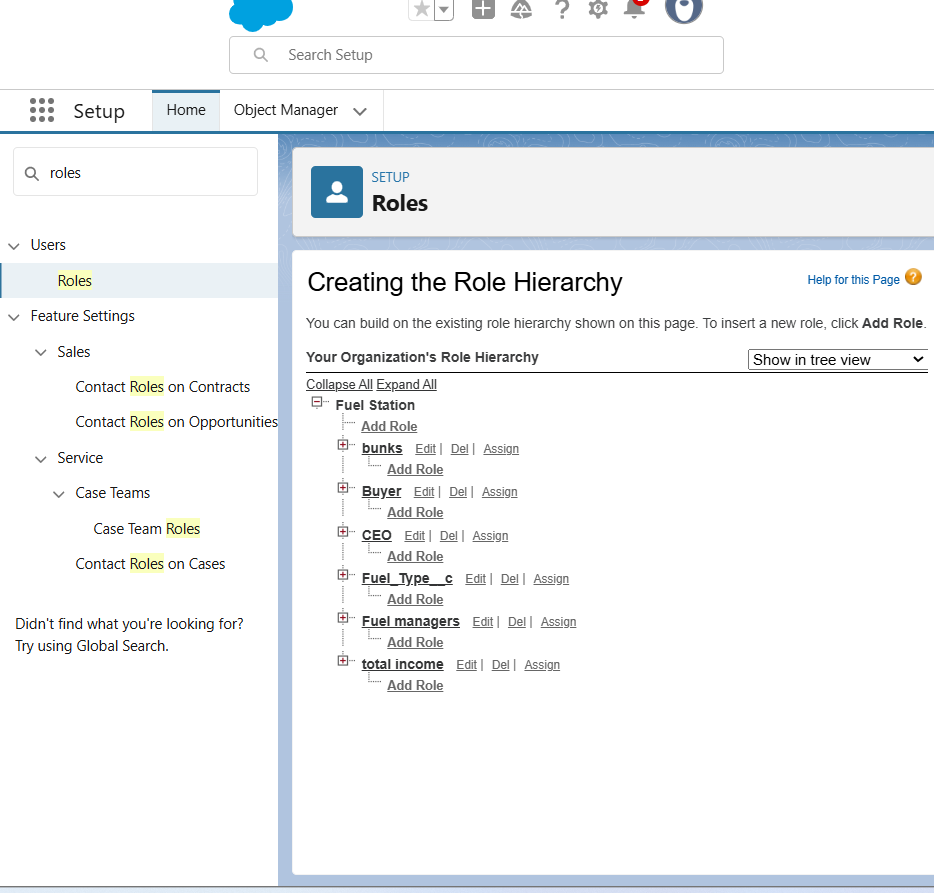
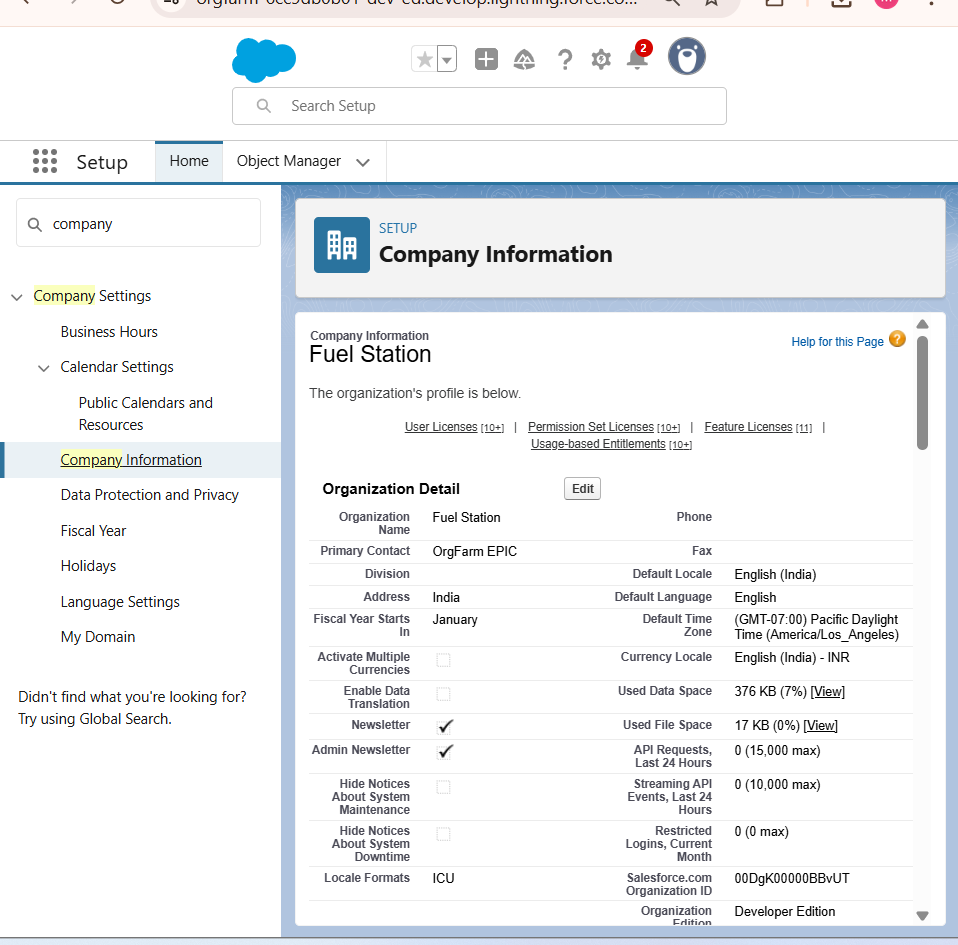
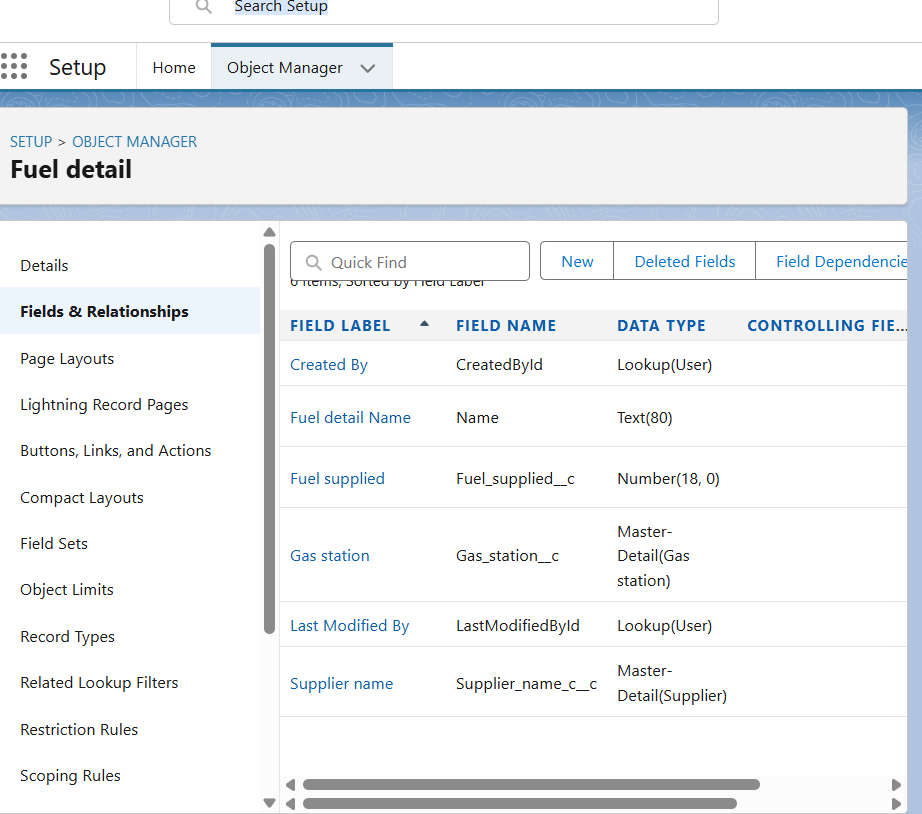
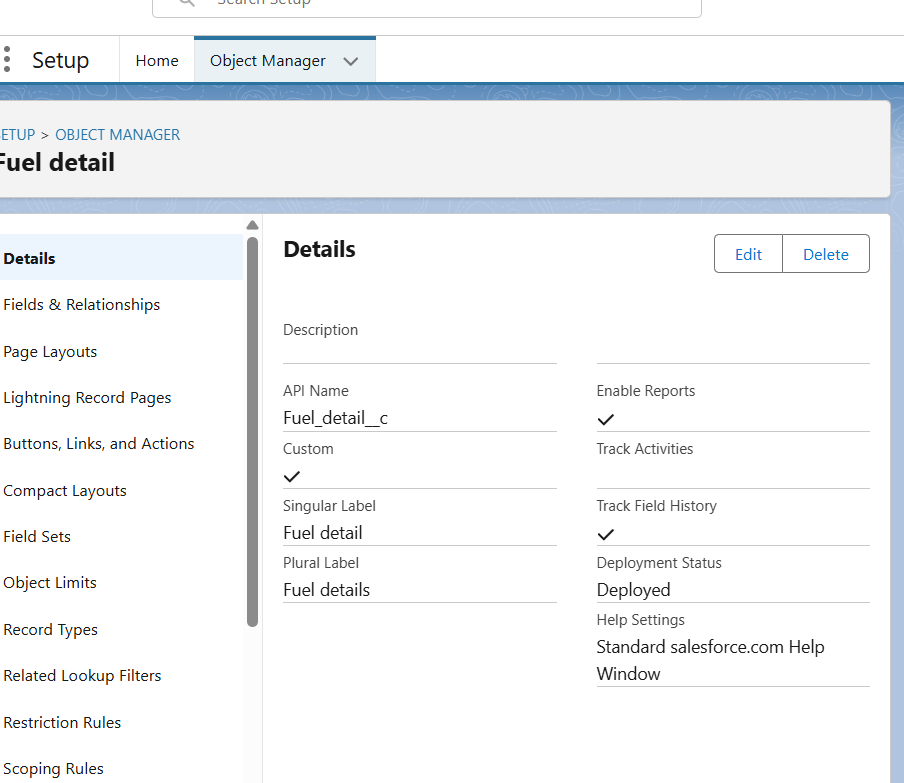
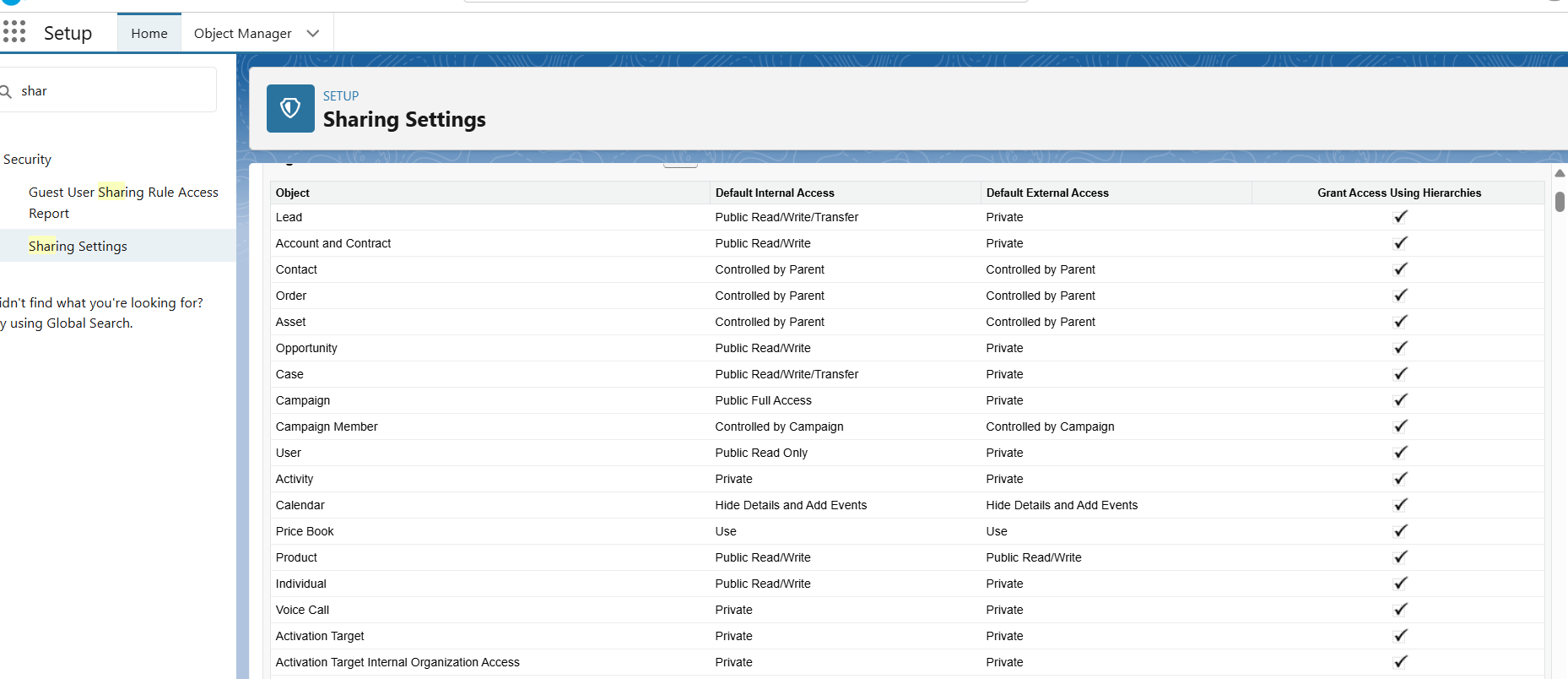
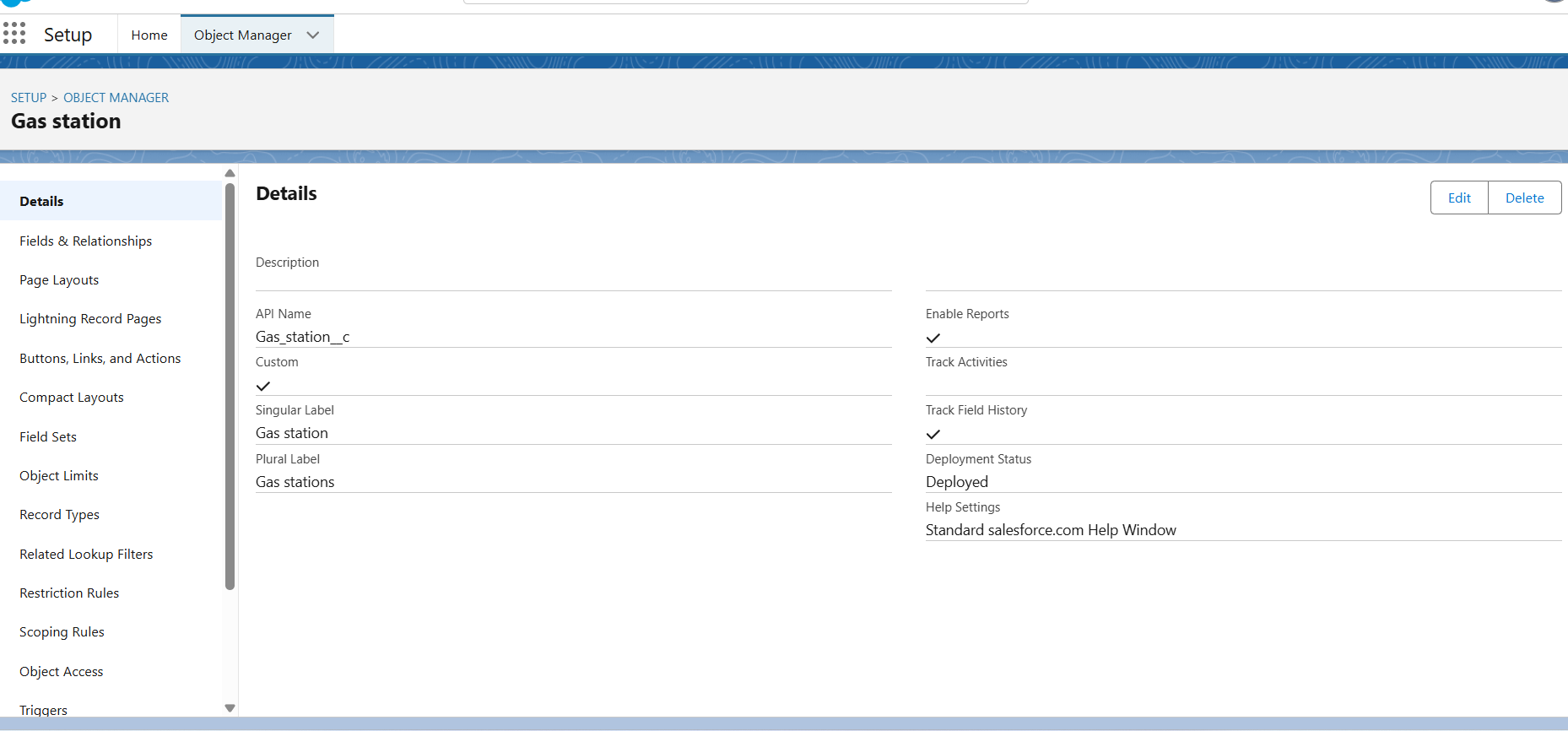
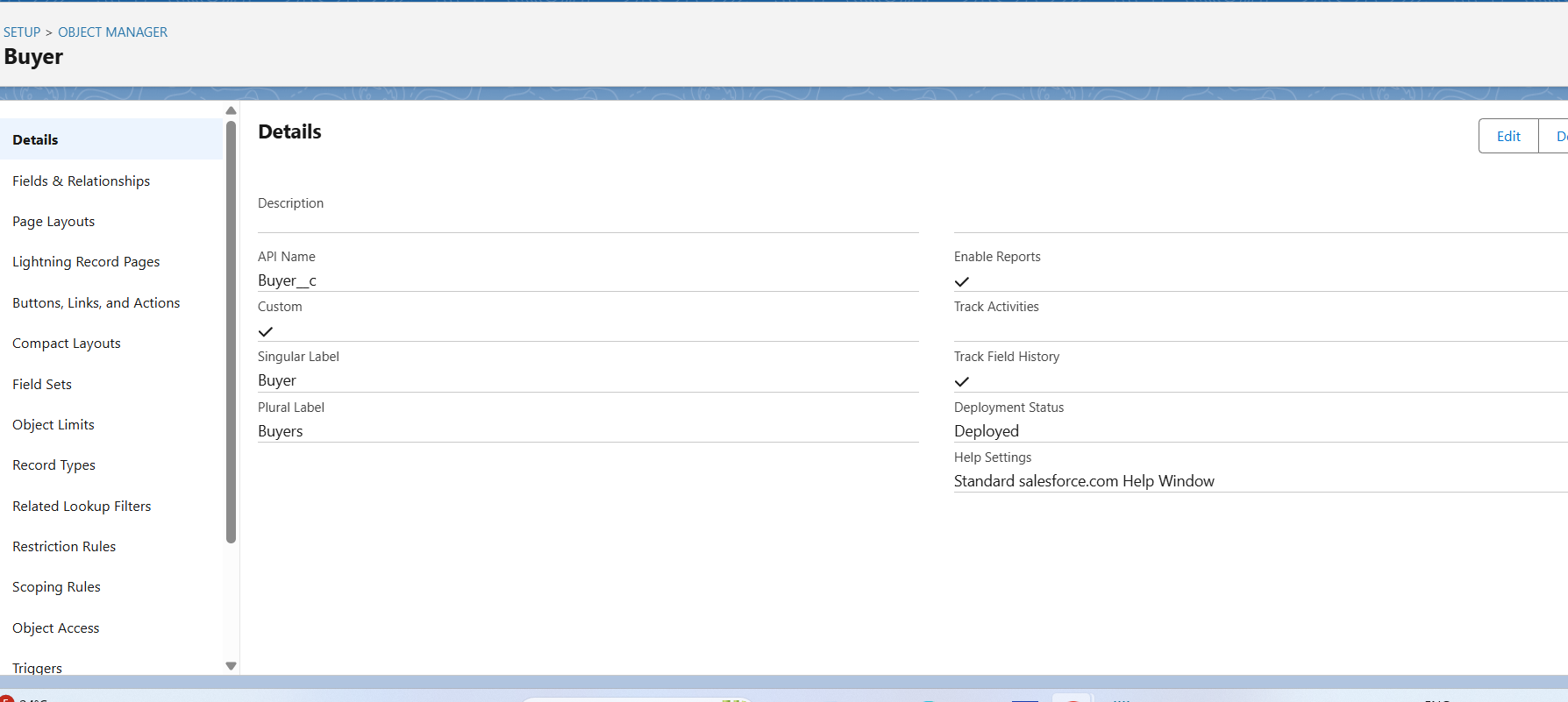
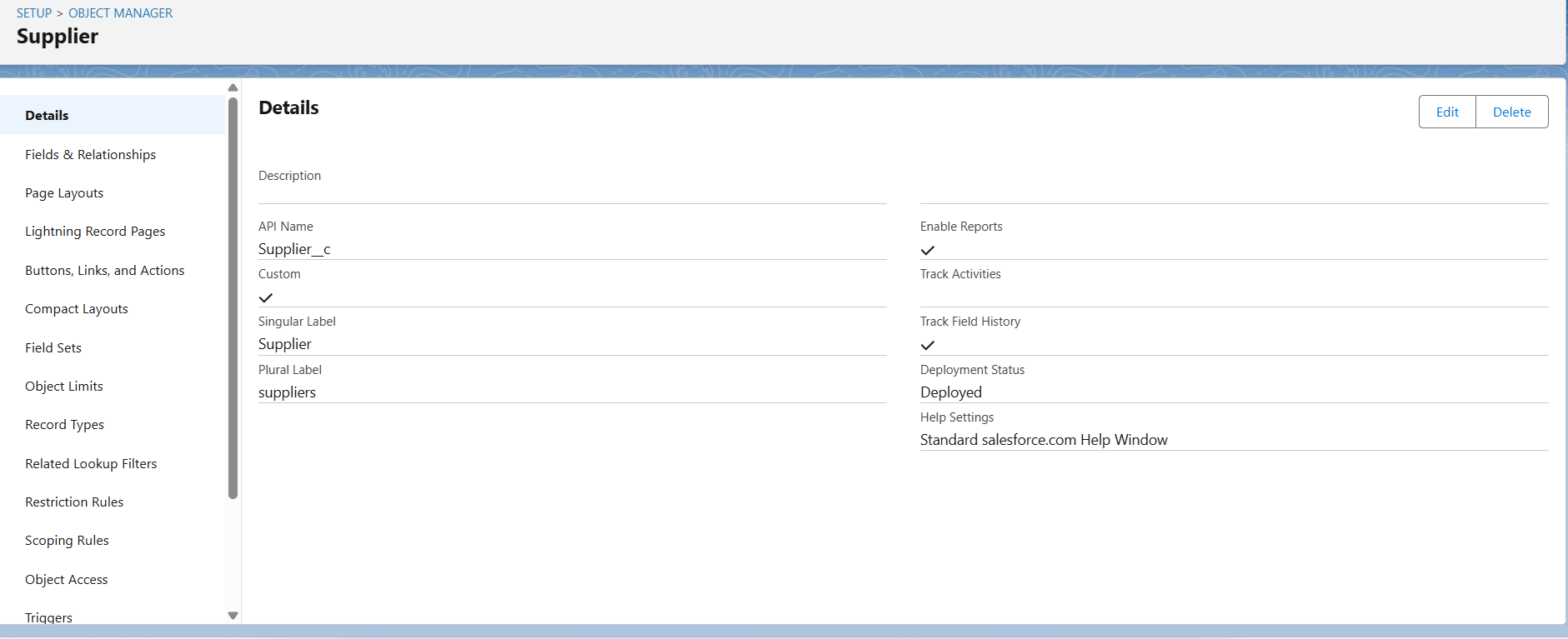
**Salesforce Edition**: Selecting Sales Cloud Developer Edition to provide flexibility for customization, testing, and scalability. This edition serves as a robust foundation for building, experimenting, and deploying CRM functionalities.

**Company Profile**: Setting up company details, including regional settings, working hours, and official holidays. This ensures system processes like reporting and scheduling align with organizational operations.

**User Setup**: Creating user accounts for Admins, Suppliers, Gas Station Managers, and Buyers. Each account is configured with permissions suited to the role, ensuring secure and efficient task execution.

**Roles & Profiles**: Establishing a structured hierarchy where Admins have complete oversight, Suppliers manage deliveries, Station Managers oversee daily activities, and buyers access loyalty and purchase features. This structure ensures accountability and clarity.

**Security:** Configuring Organization-Wide Defaults (OWD), Sharing Rules, and Login Hours. These measures protect sensitive data and ensure that access is role-appropriate and compliant with security standards.



✅ Outcome: Outcome: A secure, well-configured Salesforce org ready for customization.

# Phase 3 – Data Modeling & Relationships

Objective: Build a scalable CRM data structure for suppliers, stations, buyers, and transactions.

• **Custom Objects**: Gas Station, Supplier, Buyer, Fuel Details, Transactions.

• **Relationships**: Supplier → Fuel Details (Master-Detail), Gas Station → Fuel Details (Lookup), Buyer → Transactions (Lookup).

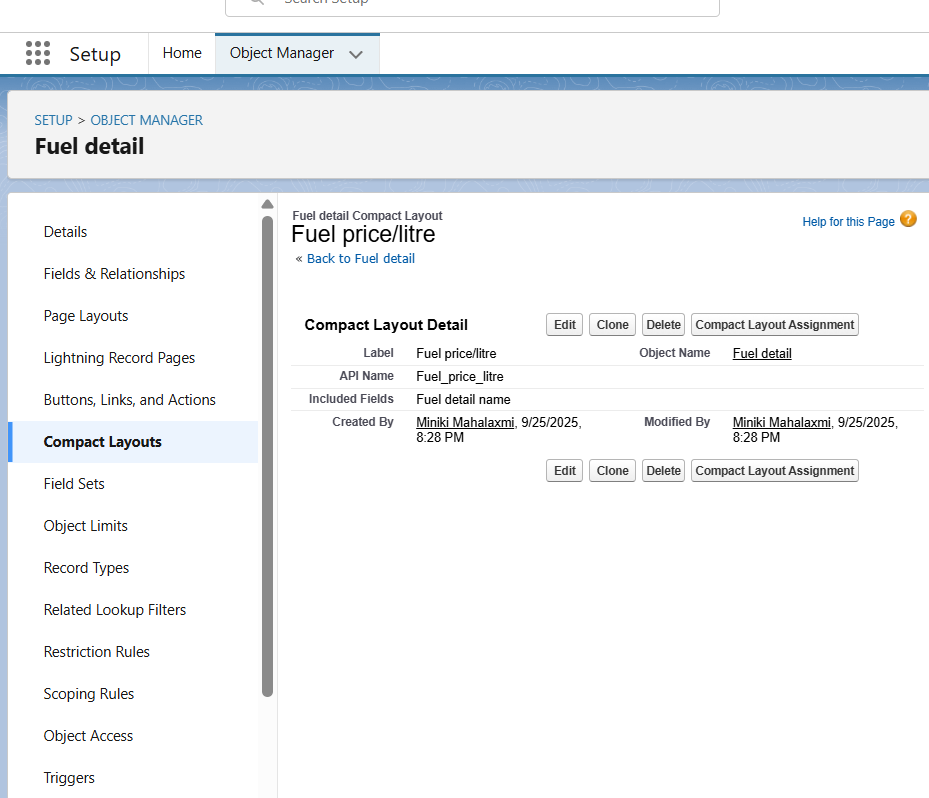
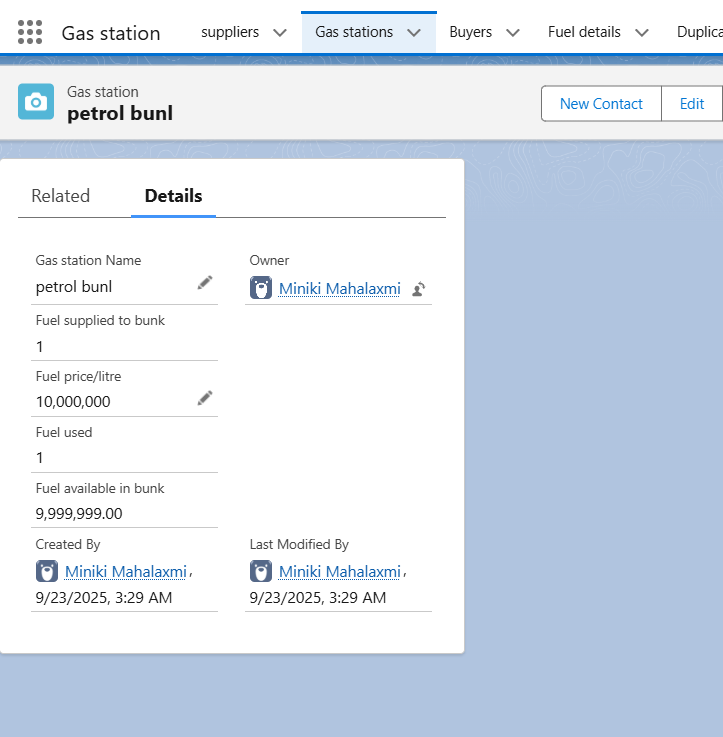
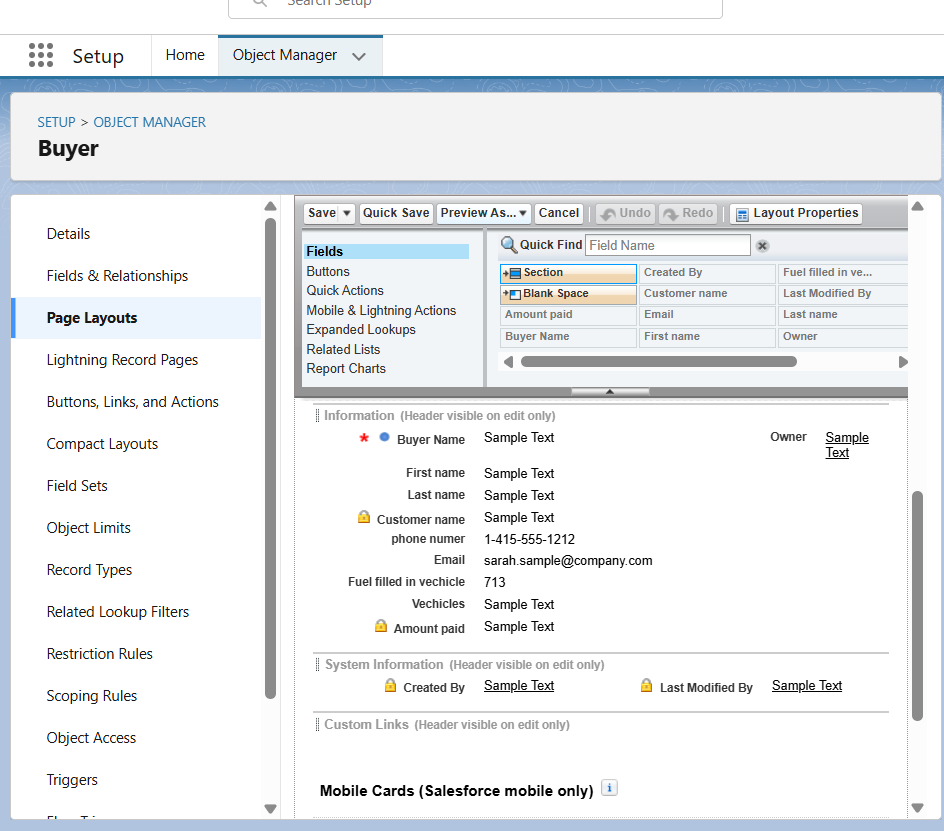
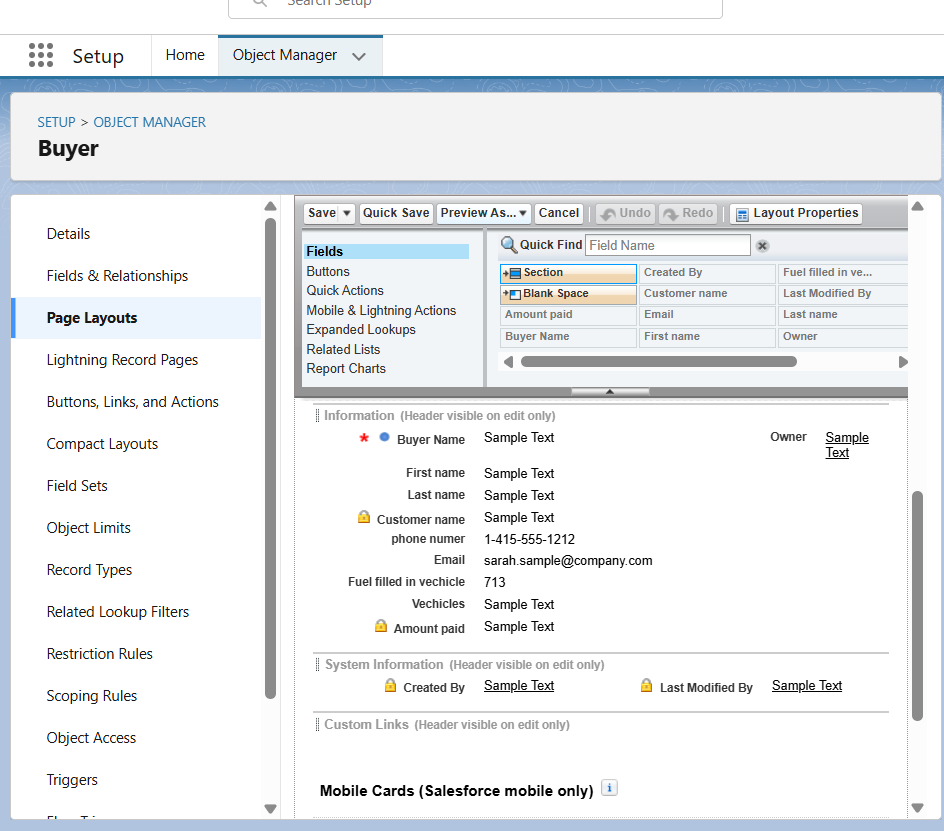
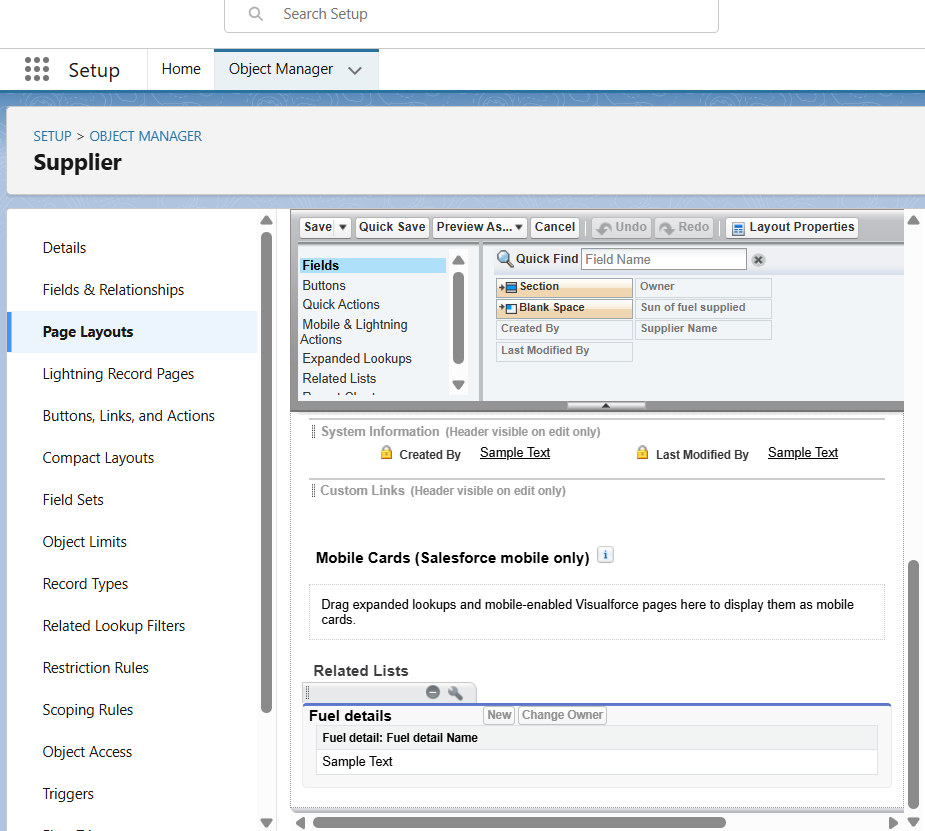
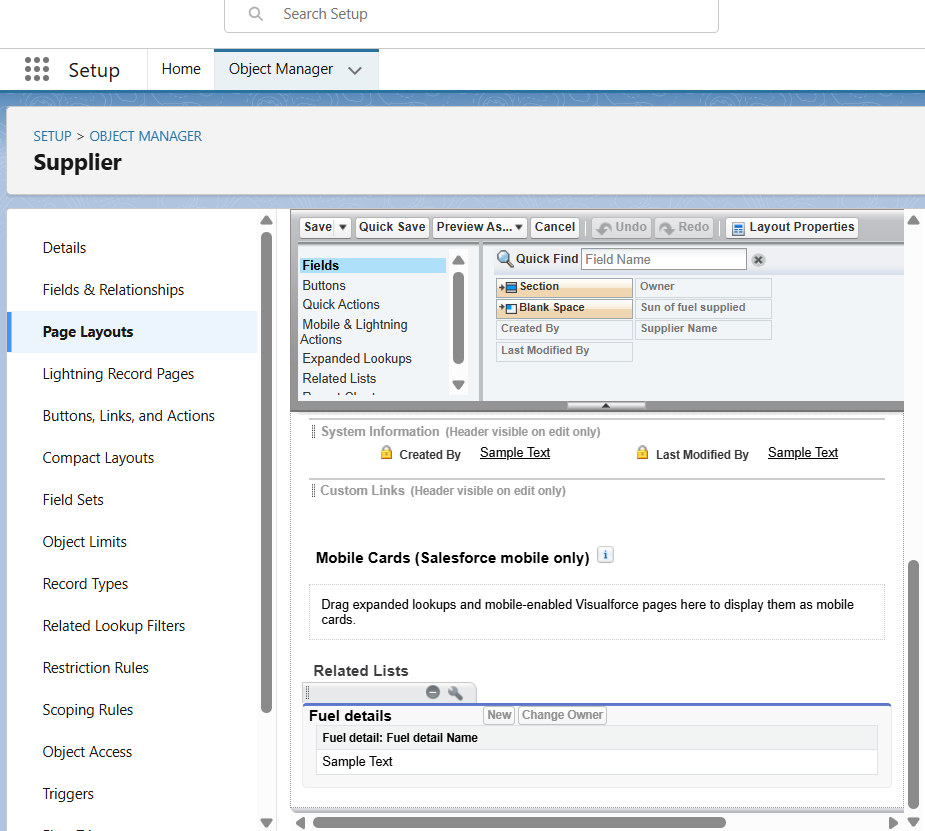
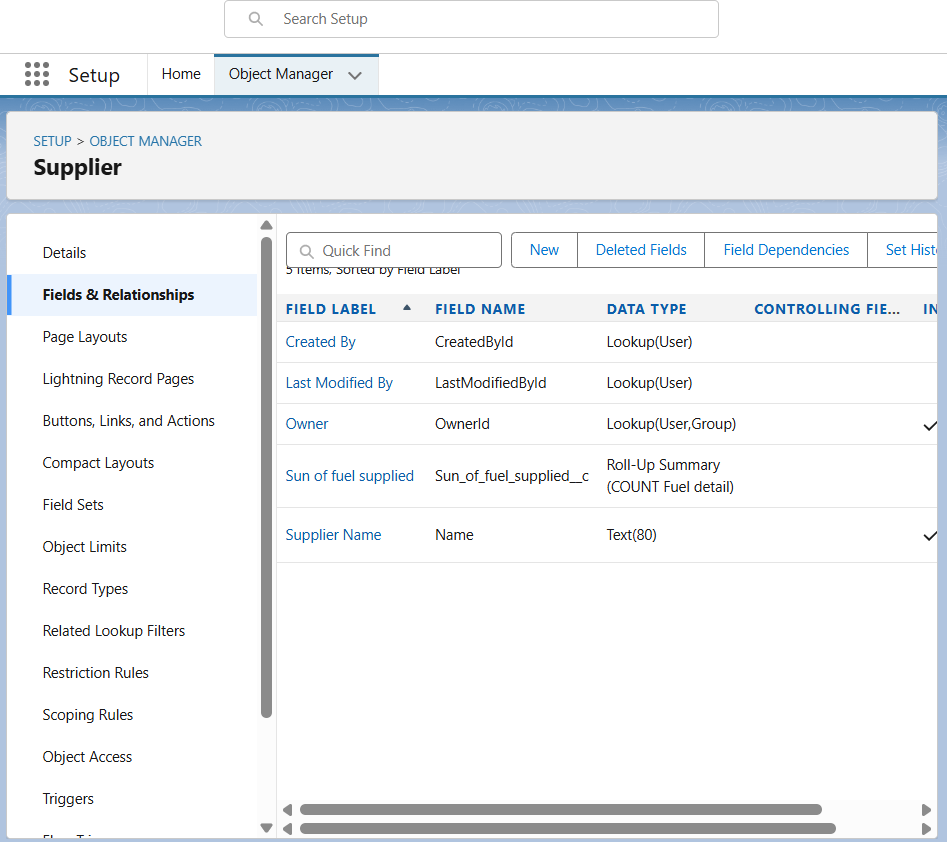
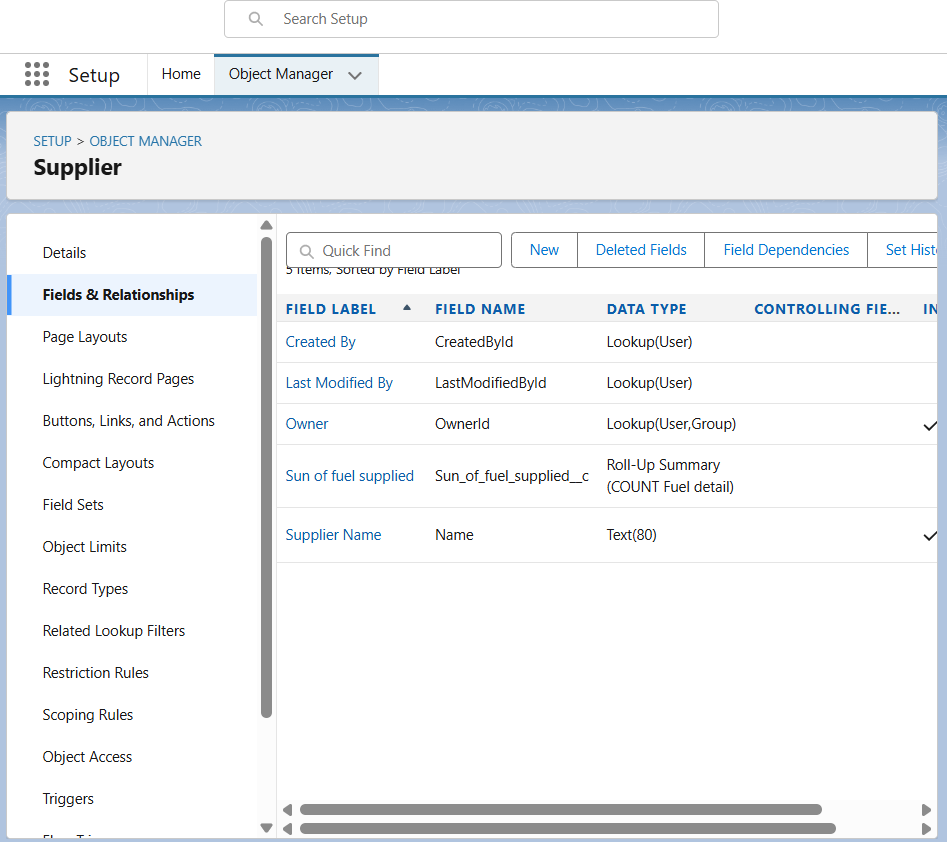
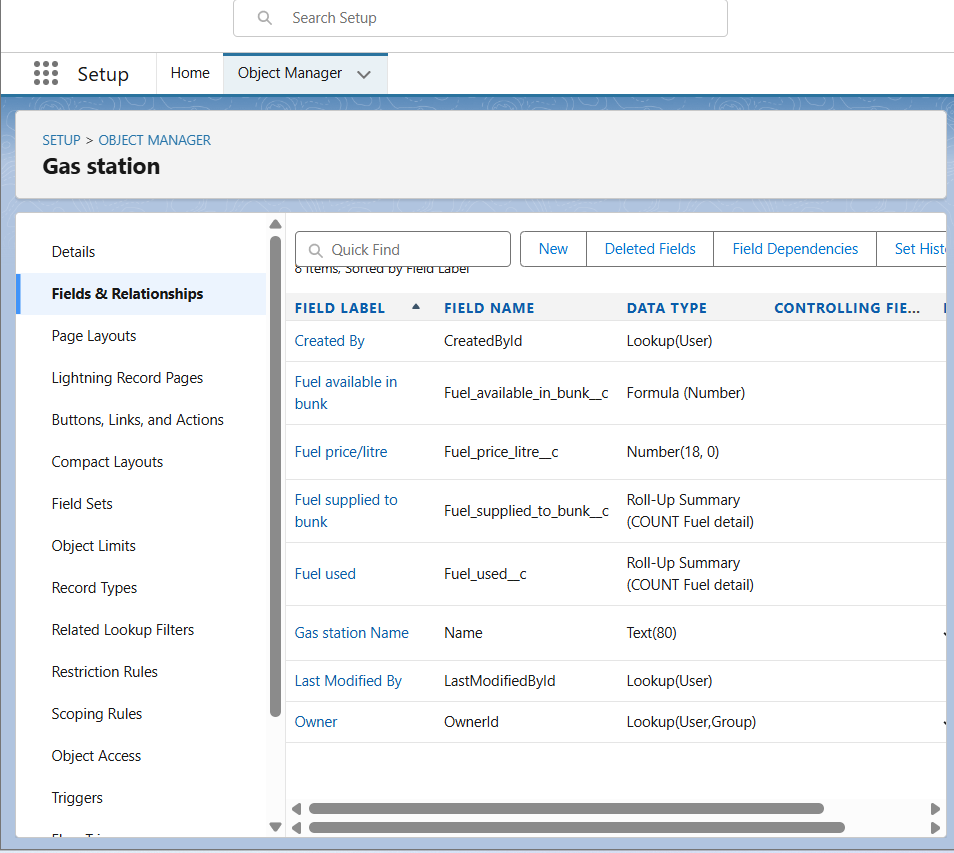
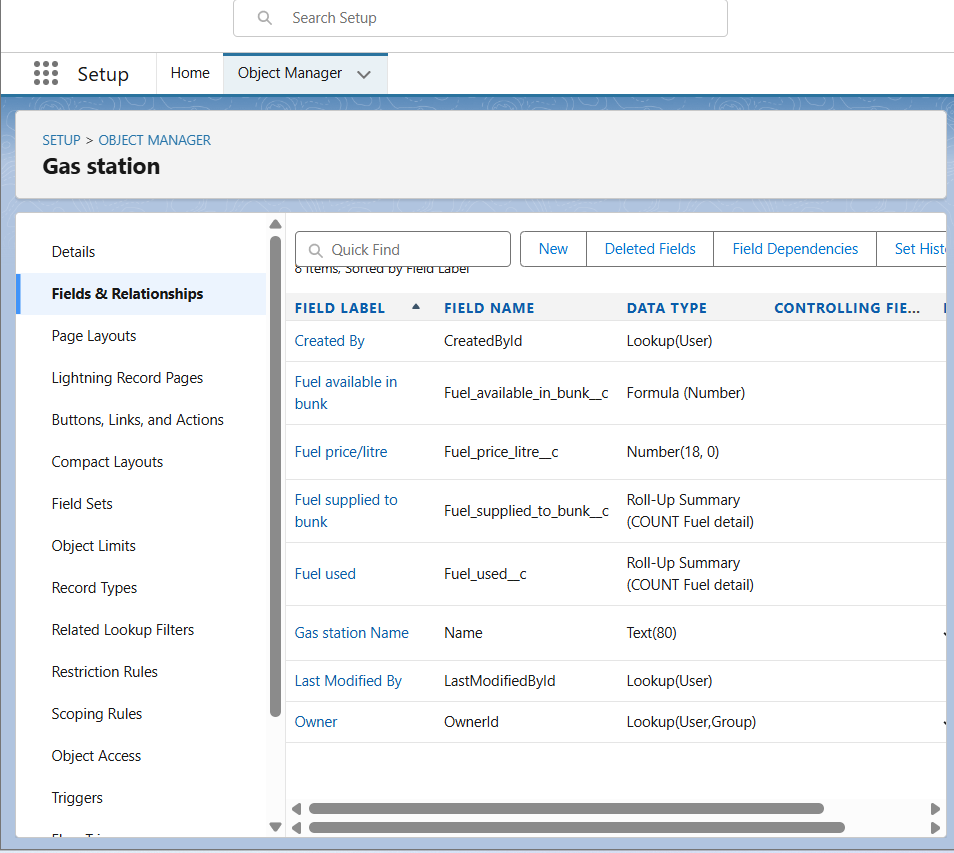
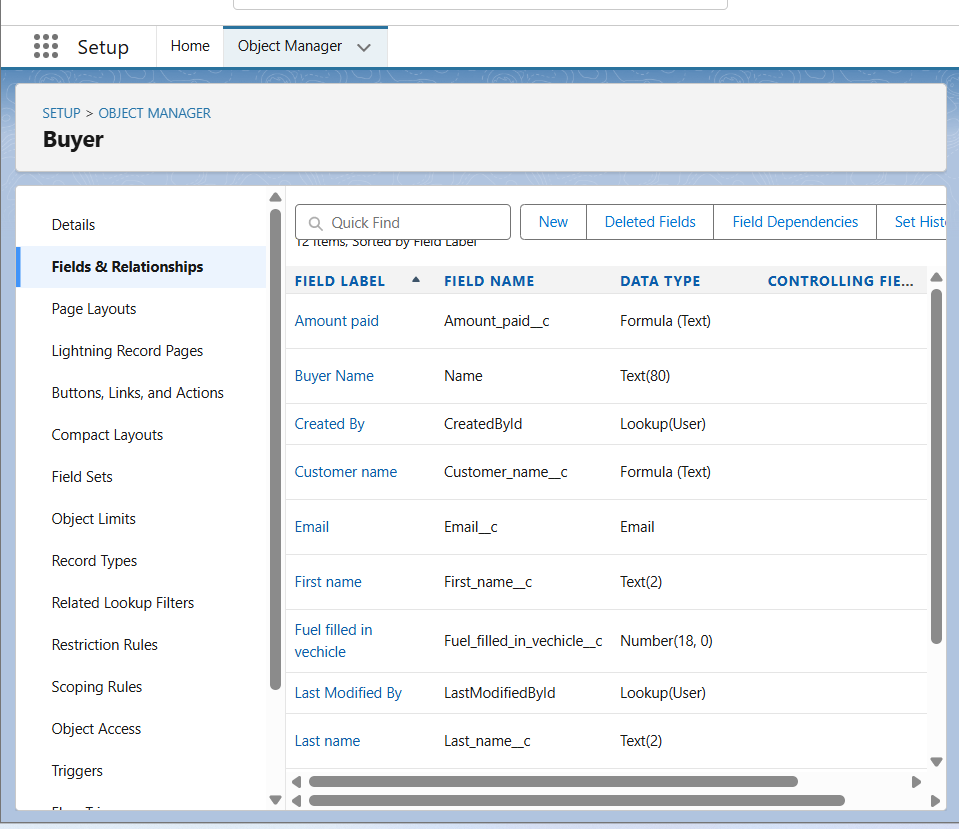
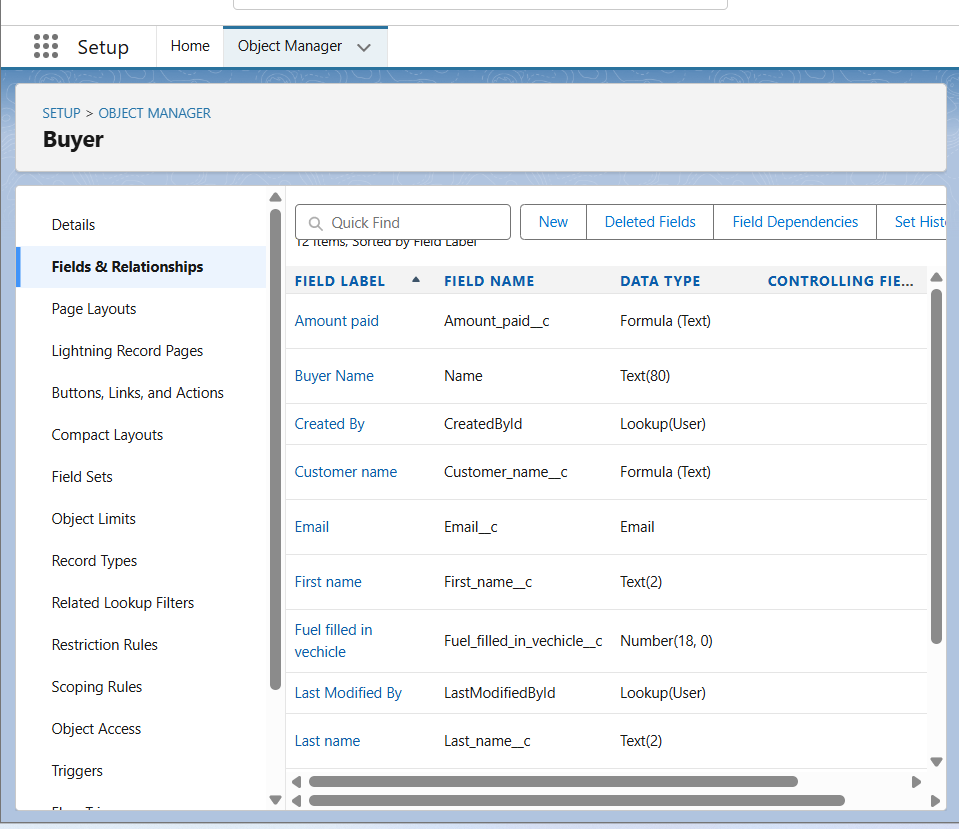
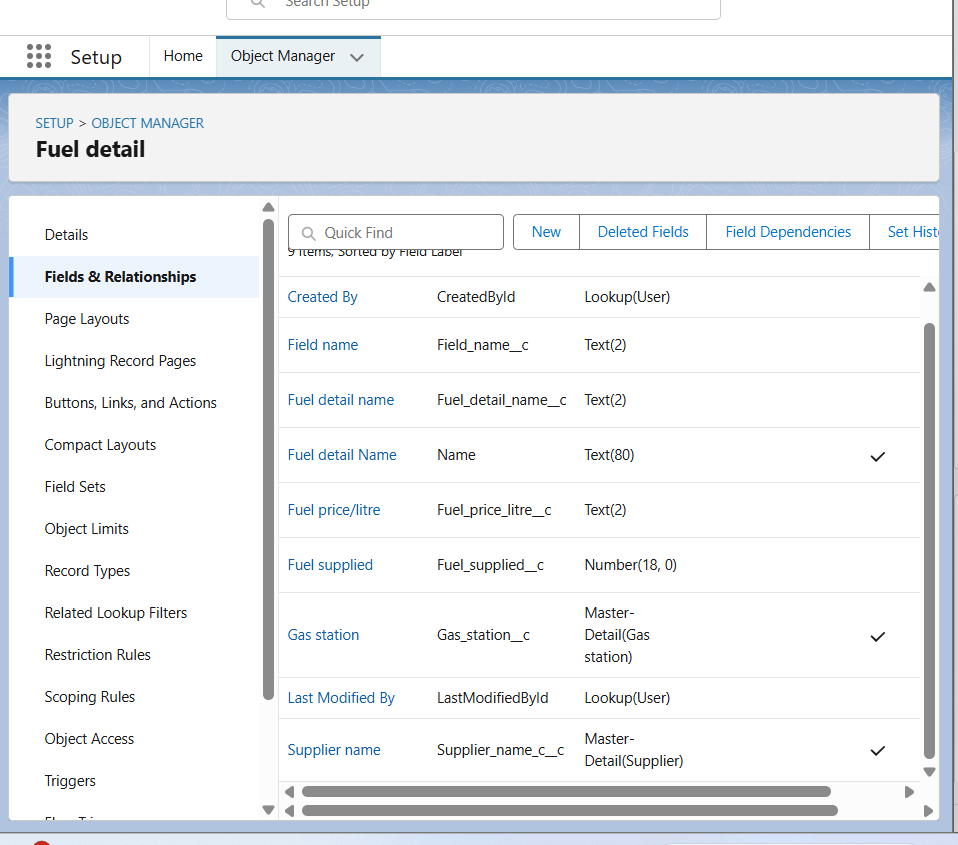
• **Fields:** Fuel Type, Quantity, Price, Date Supplied; Transactions → Amount, Buyer, Fuel Type, Purchase Date.

• **Layouts & Record Types**: Configure layouts for Gas Station and Buyer for better usability.

• **Schema Builder**: Relationships for various data

• **Junction objects**: Had multiple data suppliers

• **Compact layouts:** status, fuel details we added compact layouts



✅ Outcome: Outcome: A robust Salesforce data model enabling accurate fuel station operations tracking.

# Phase 4 – Process Automation (Admin)

Objective: Automate critical business workflows to improve efficiency.

• **Validation Rules:** Ensure Amount > 0, Fuel Quantity > 0.

• **Approval Processes:** Automate high-value supplier contract approvals.

• **Flows:** Automatically update stock after transactions.

• **Email Alerts:** Notify managers for low stock or pending approvals.

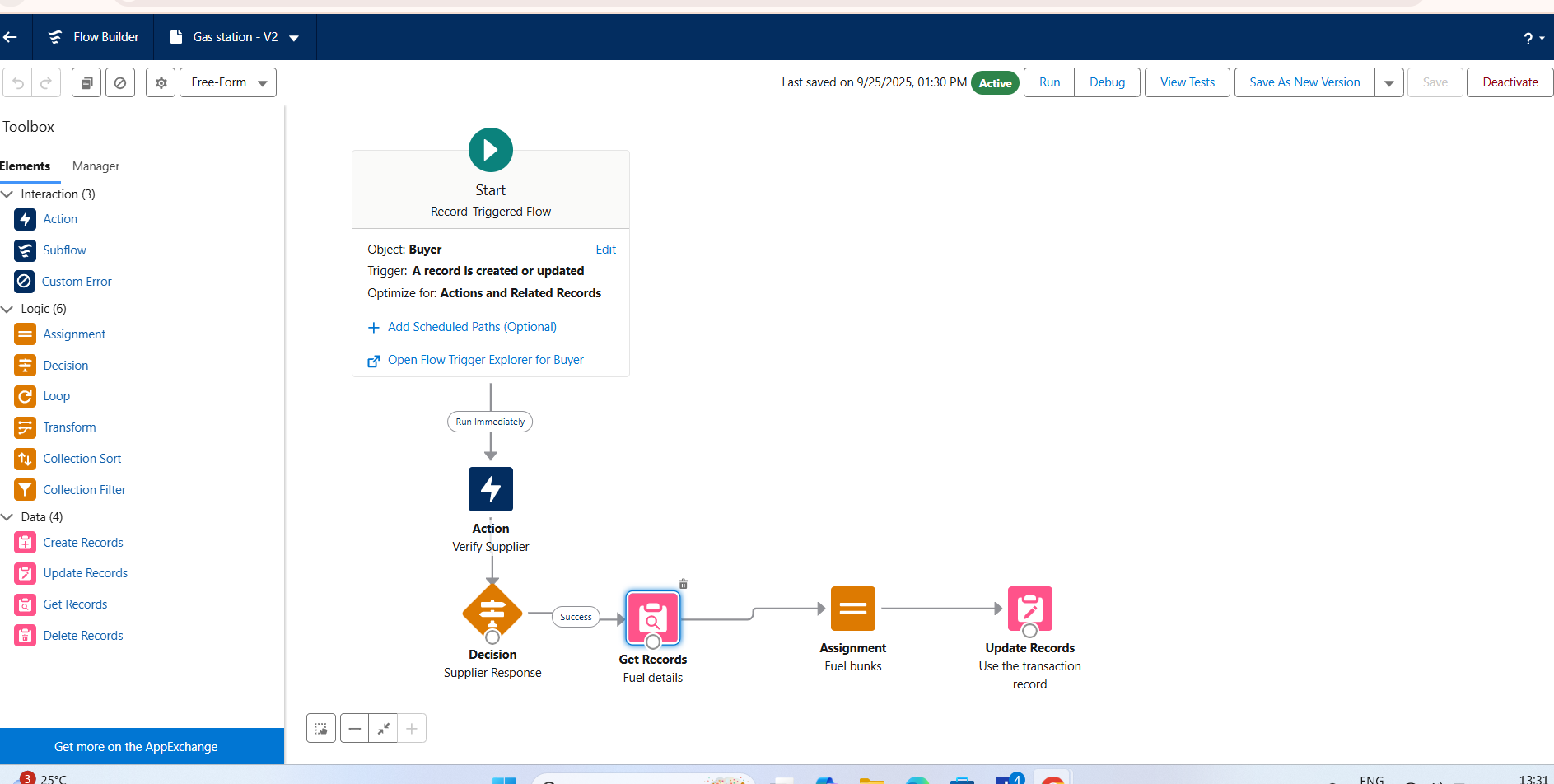
• **Reminders:** Send notifications for supplier reorders.

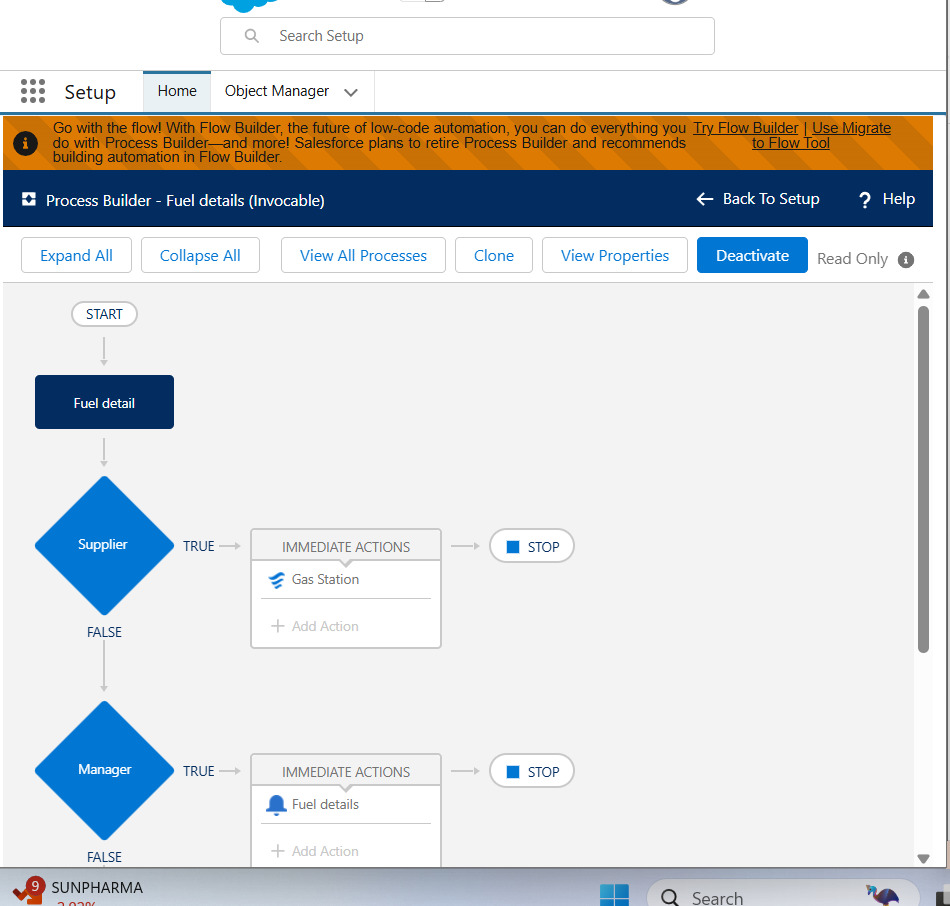
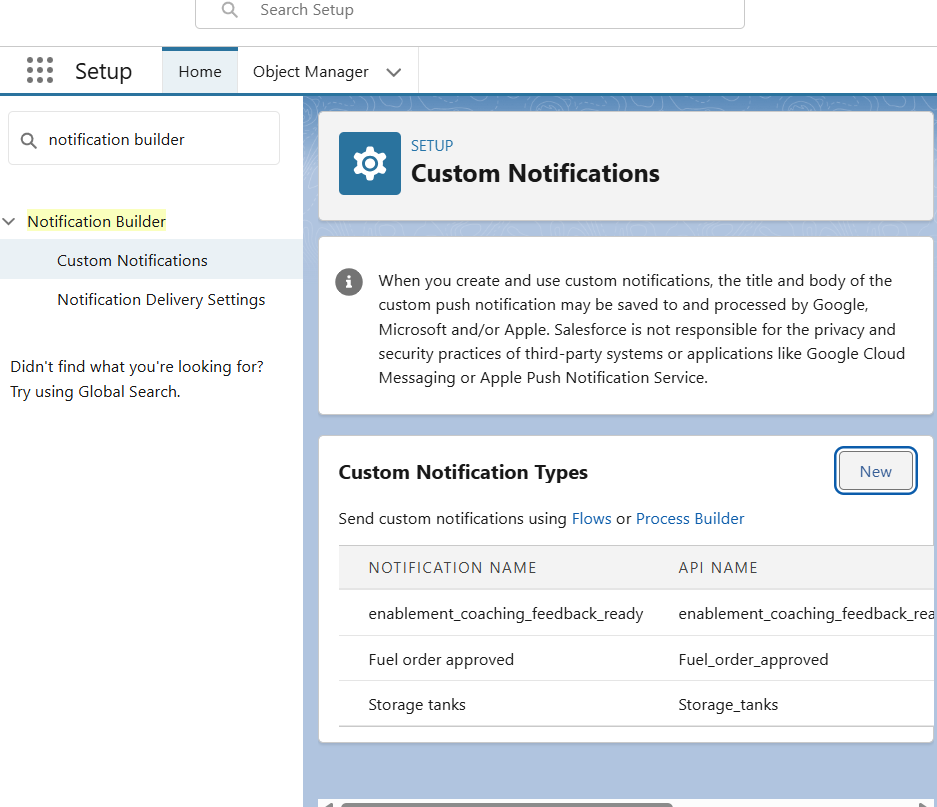
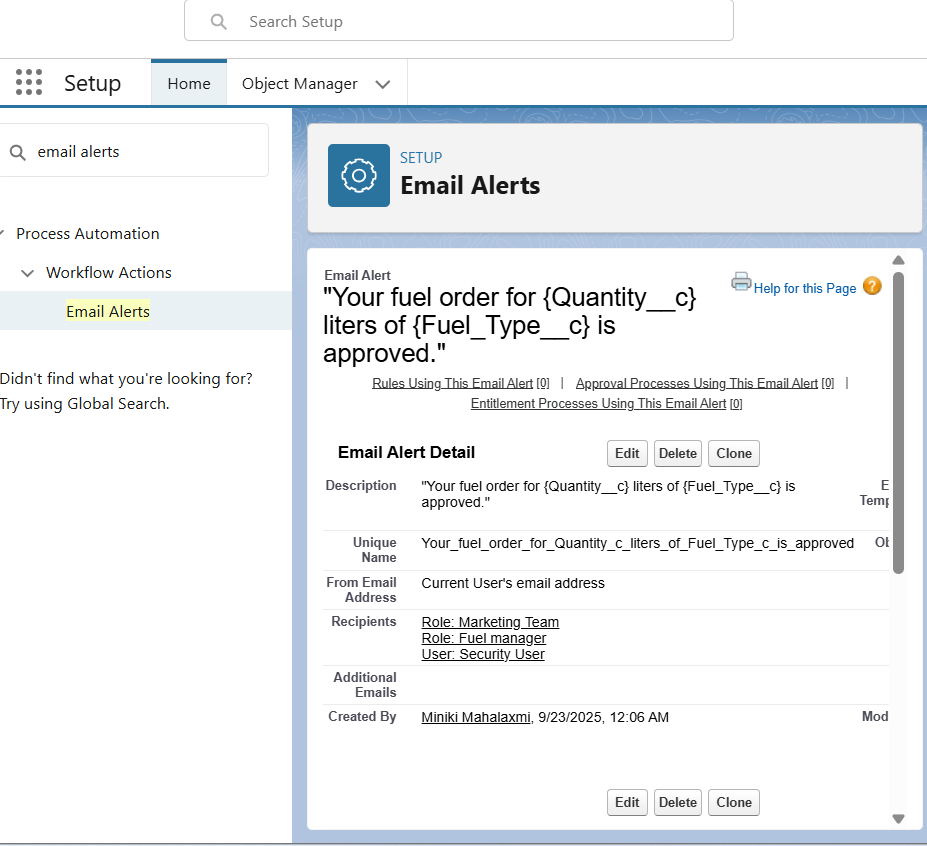
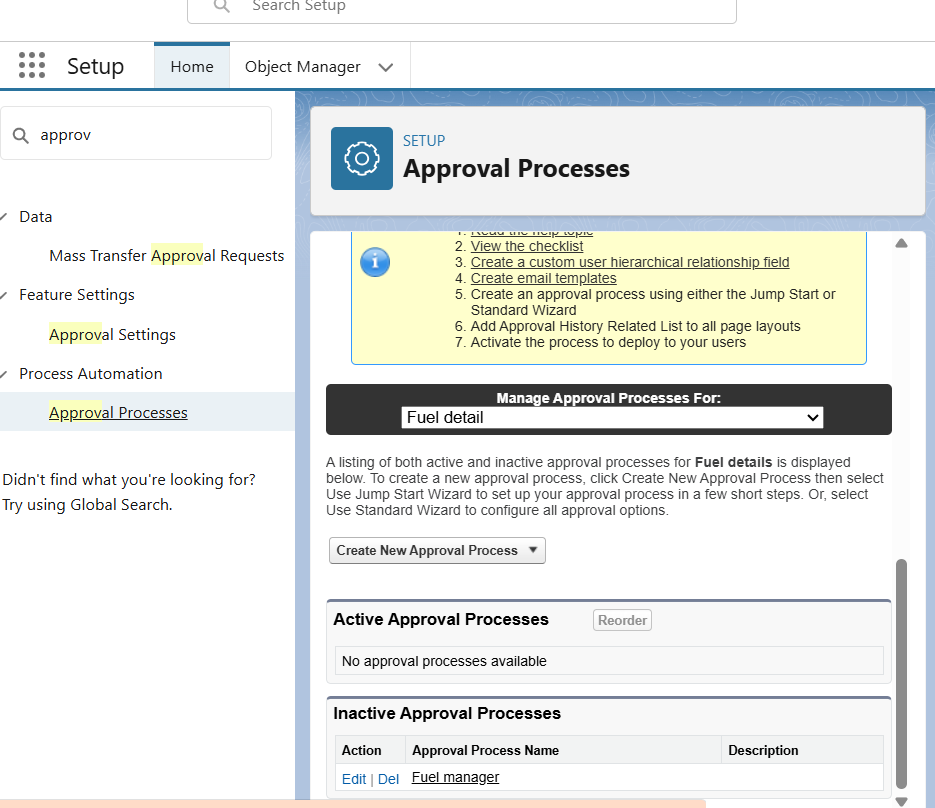
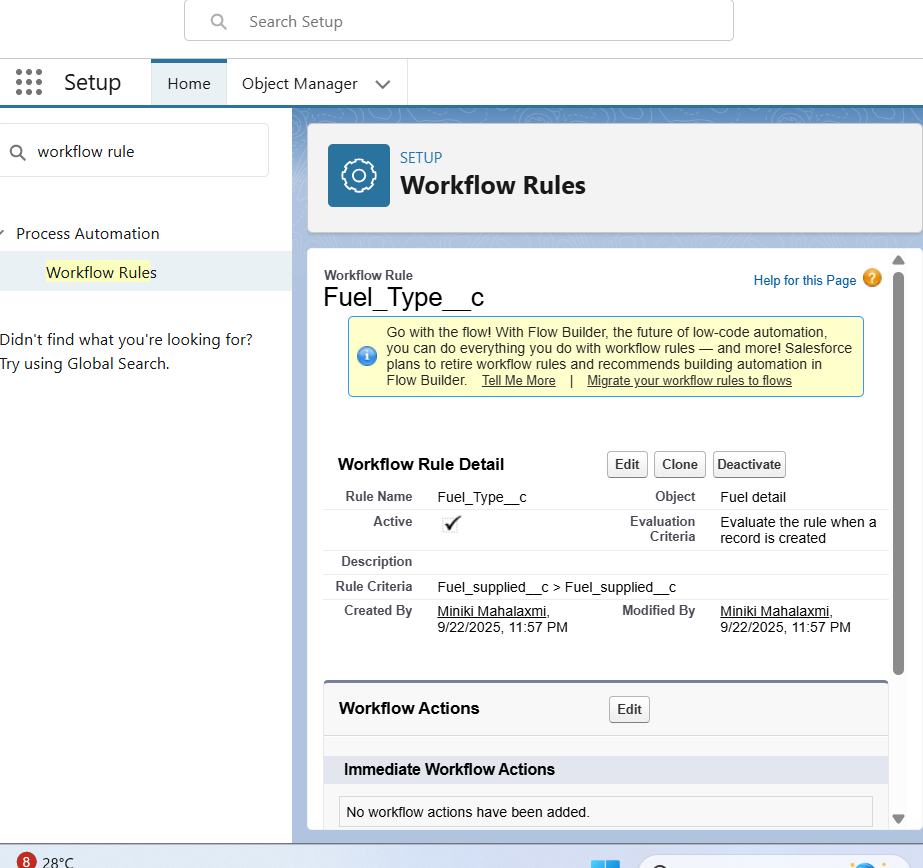
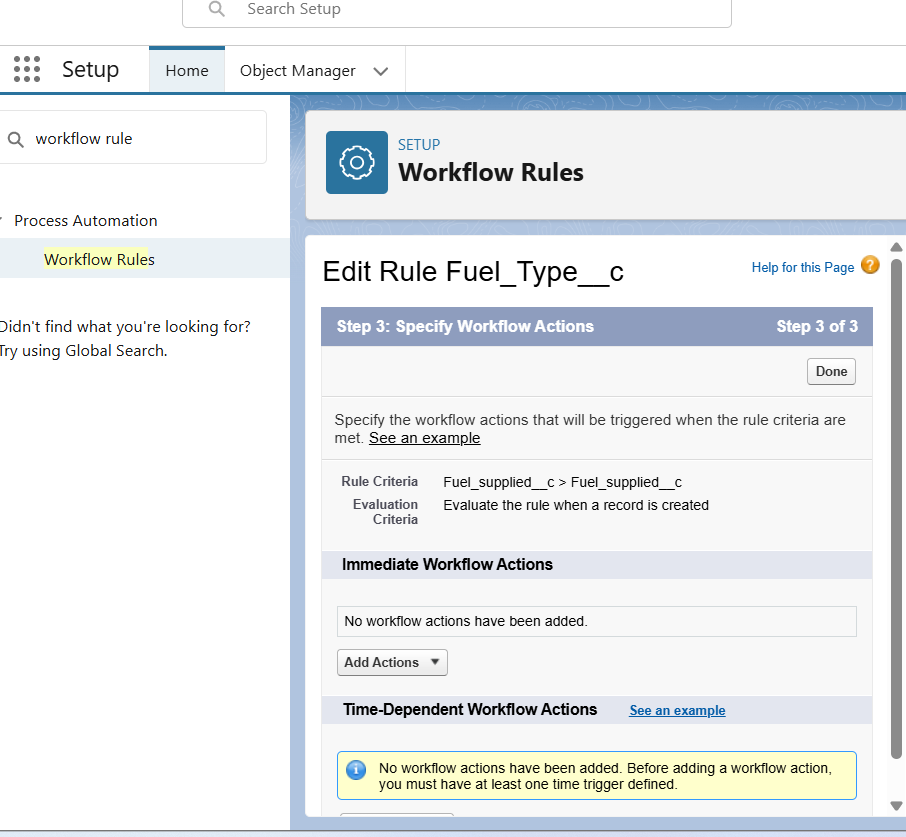
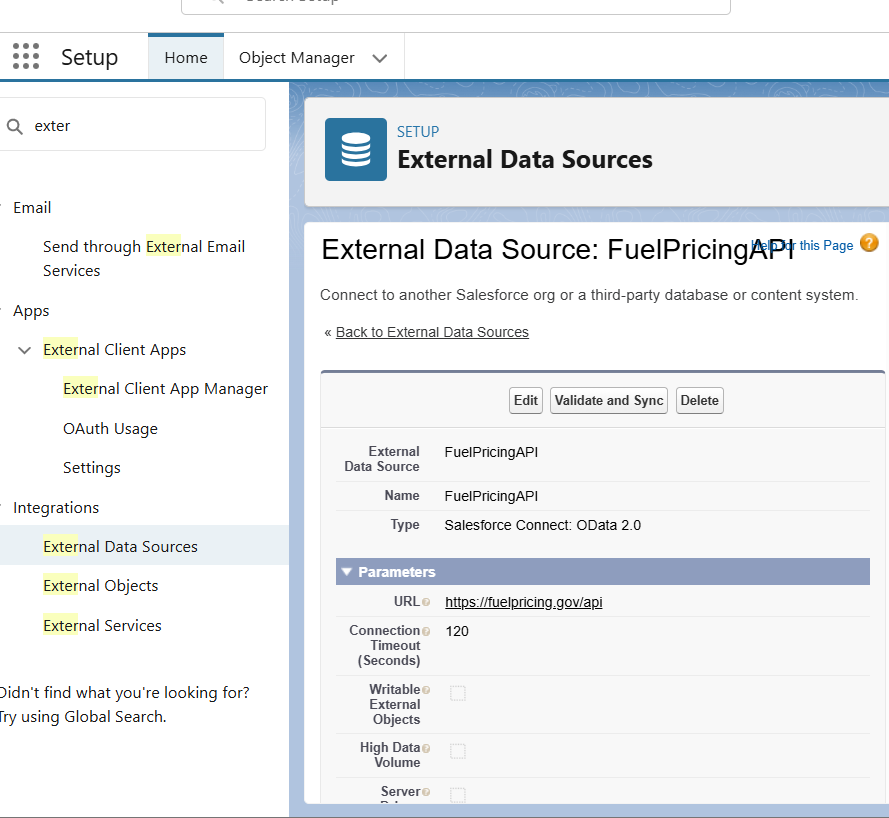
• **Custom notification:** Set up notification for agent to update fuel details.

• **Field updates:** After approval Fuel manager =” confirmed”

• **Workflow rules:** Auto-send email when buyer send bulk amount

• **Workflow rules:** Replaced by flow now





✅ Outcome: Outcome: Reduced manual work with automated and reliable business processes.

# Phase 5 – Apex Programming (Developer)

Objective: Implement advanced business logic with Salesforce Apex.

• **Triggers:** Prevent duplicate fuel supply entries.

trigger DeliveryTrigger on Delivery\_\_c (before insert, before update) {  
 DeliveryHandler.beforeSave(Trigger.new, Trigger.oldMap);  
 }  
   
public class DeliveryHandler {  
 public static void beforeSave(List<Delivery\_\_c> newList, Map<Id, Delivery\_\_c> oldMap) {  
 // custom validation logic here  
 }  
 }

• **SOQL:** Retrieve fuel stock and transaction details efficiently.

List<Delivery\_\_c> deliveries = [SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c WHERE Delivery\_Status\_\_c = 'Pending'];  
 SOSL – Search Across Customers and OrdersList<List<SObject>> results =   
 [FIND 'Kavya\*' IN ALL FIELDS   
 RETURNING Customer\_\_c(Id, Name), Order\_\_c(Id, Status\_\_c)];

• **Batch Apex:** Run nightly batch jobs for sales and fuel stock updates.

global class UpdateDeliveryBatch implements Database.Batchable<SObject> {  
 global Database.QueryLocator start(Database.BatchableContext bc) {  
 return Database.getQueryLocator('SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c');  
 }  
 global void execute(Database.BatchableContext bc, List<Delivery\_\_c> scope) {  
 for (Delivery\_\_c d : scope) {  
 d.Delivery\_Status\_\_c = 'Archived';  
 }  
 update scope;  
 }  
 global void finish(Database.BatchableContext bc) {}  
 }

• **Future Methods:** Enable asynchronous calls for supplier APIs.

• **Test Classes:** Ensure code reliability and maintain test coverage.

@isTest  
 private class DeliveryTriggerTest {  
 @isTest static void testDuplicateDeliveryNumber() {  
 // insert deliveries and check addError  
 }  
 }

• **Apex Triggers:** prevent overlapping bookings for the same Fuel.

trigger DeliveryTrigger on Delivery\_\_c (before insert, before update) {  
 for (Delivery\_\_c d : Trigger.new) {  
 Boolean exists = [  
 SELECT Id FROM Delivery\_\_c   
 WHERE Delivery\_Number\_\_c = :d.Delivery\_Number\_\_c   
 AND Id != :d.Id  
 LIMIT 1  
 ] != null;  
 if (exists) {  
 d.addError('This Delivery Number already exists.');  
 }  
 }  
 }

• **Classes & Objects:** Create a BookingService class for reusable logic.

public class DeliveryService {  
 public static void confirmDelivery(Id deliveryId) {  
 Delivery\_\_c d = [SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c WHERE Id = :deliveryId];  
 d.Delivery\_Status\_\_c = 'Confirmed';  
 update d;  
 }  
 }

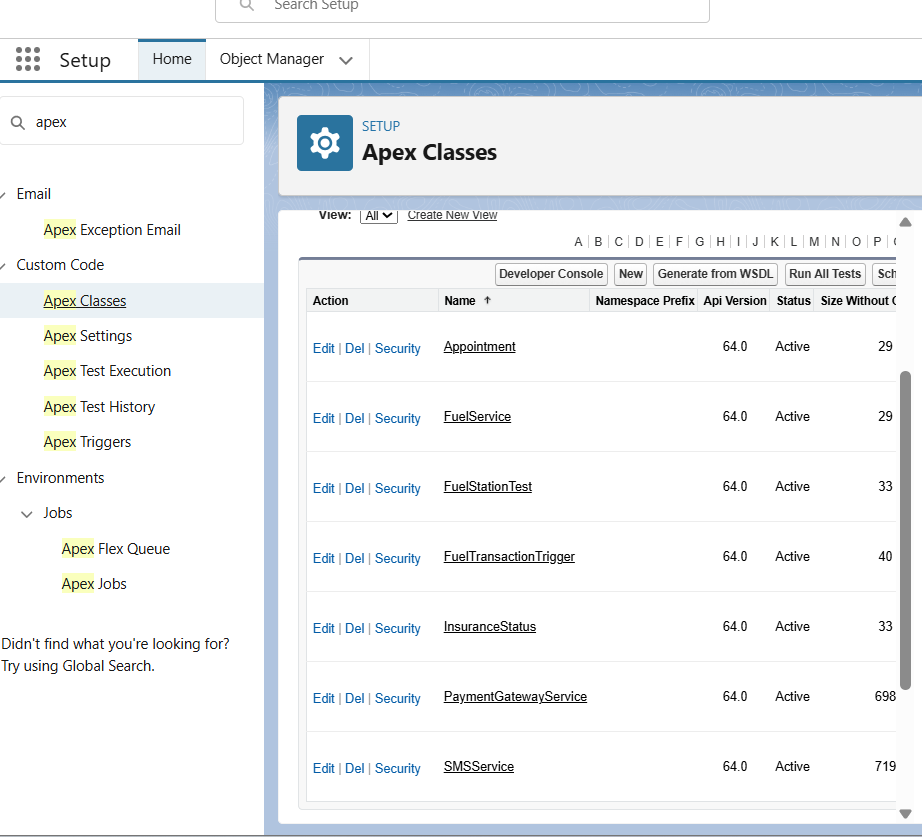
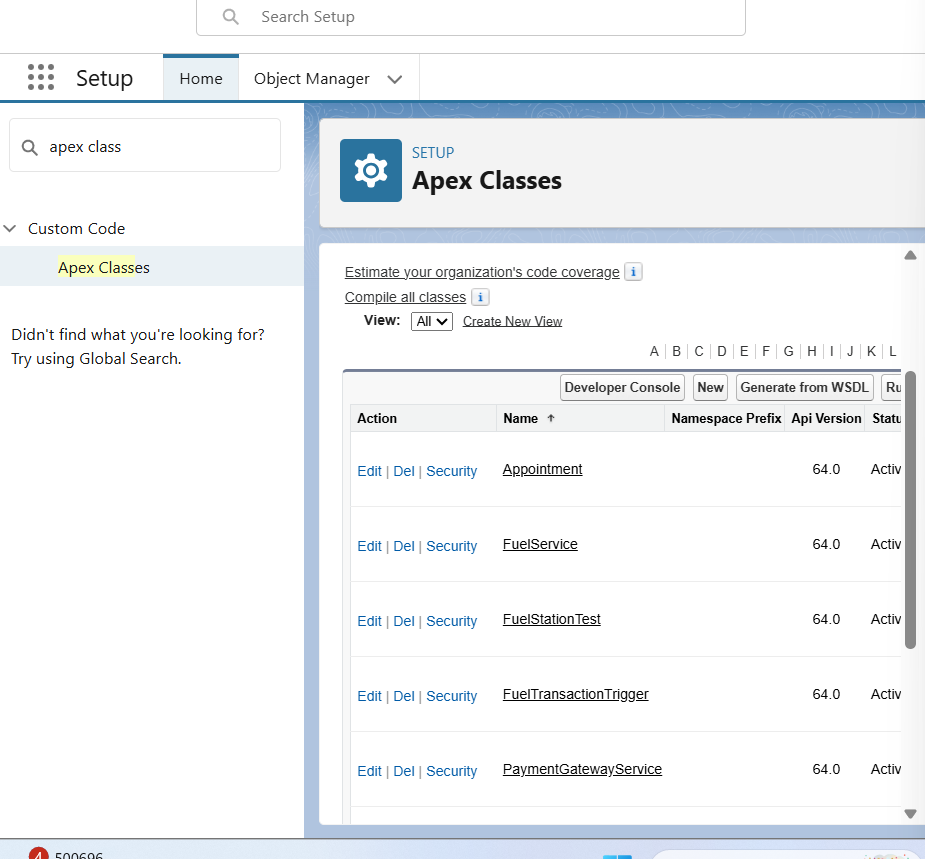
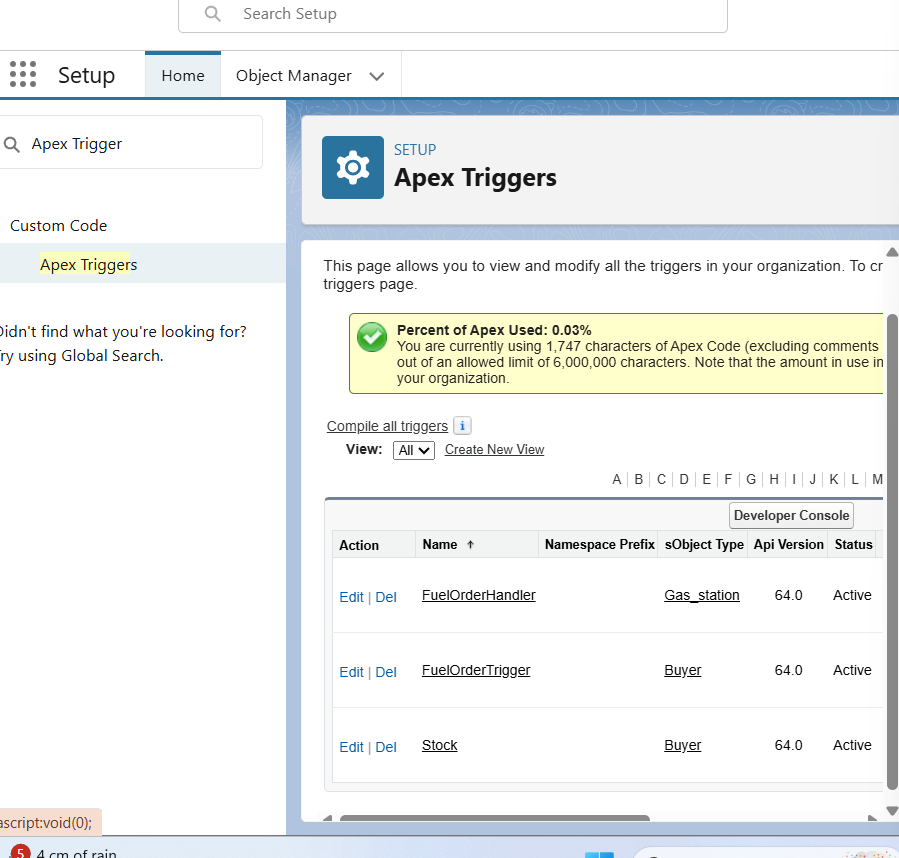
• **Collections:** Store multiple vehcile IDs in Set to avoid duplicates.

List<String> orderIds = new List<String>();  
 Set<Id> customerIds = new Set<Id>();  
 Map<Id, Order\_\_c> orderMap = new Map<Id, Order\_\_c>([SELECT Id, Status\_\_c FROM Order\_\_c]);

• **Asynchronous Processing:** Batch + Queueable + Future = async jobs.

• **Future Methods:** Call external insurance API async.

public class PaymentGateway {  
 @future(callout=true)  
 public static void processPayment(Id paymentId) {  
 // external API call for payment processing  
 }  
 }

✅ Outcome: Outcome: A scalable, high-performance backend with advanced business logic.

# Phase 6 – User Interface Development

Objective: Deliver a user-friendly CRM experience with Lightning components.

• **Lightning App:** Create 'Fuel CRM' app with a professional interface.

• **Navigation Tabs:** Gas Station, Suppliers, Buyers, Transactions, Reports.

• **Dashboards:** Visualize Fuel Sales, Supplier Deliveries, and Buyer Loyalty.

• **Lightning Web Components (LWC):** Develop a search tool for fuel availability by station/date.

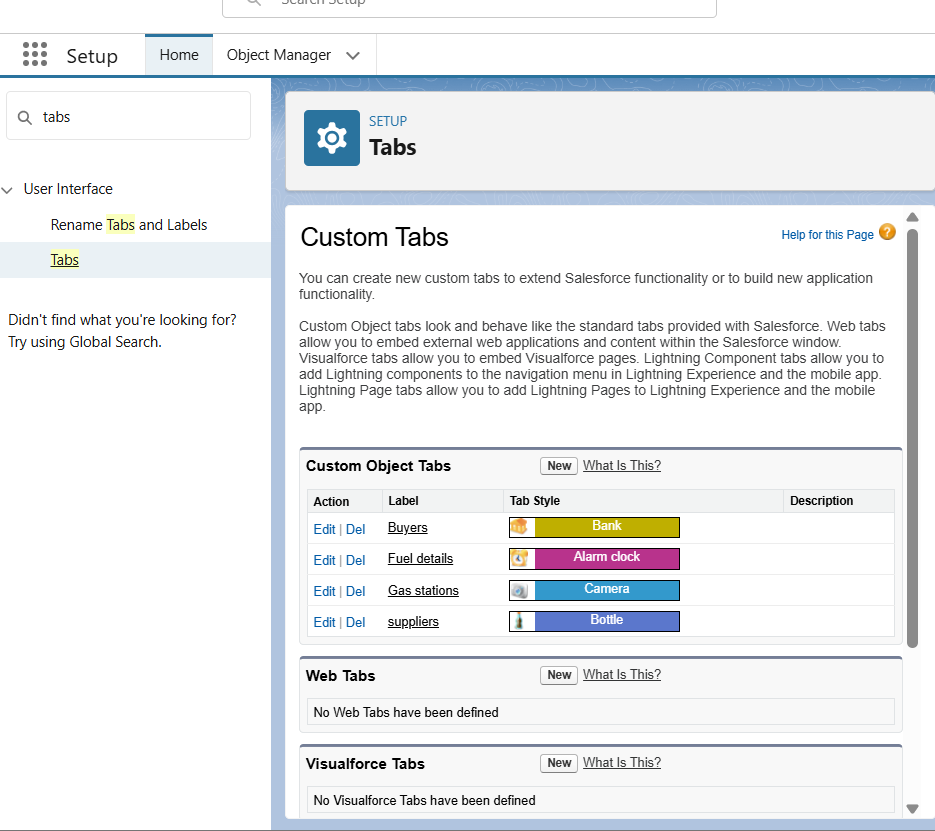
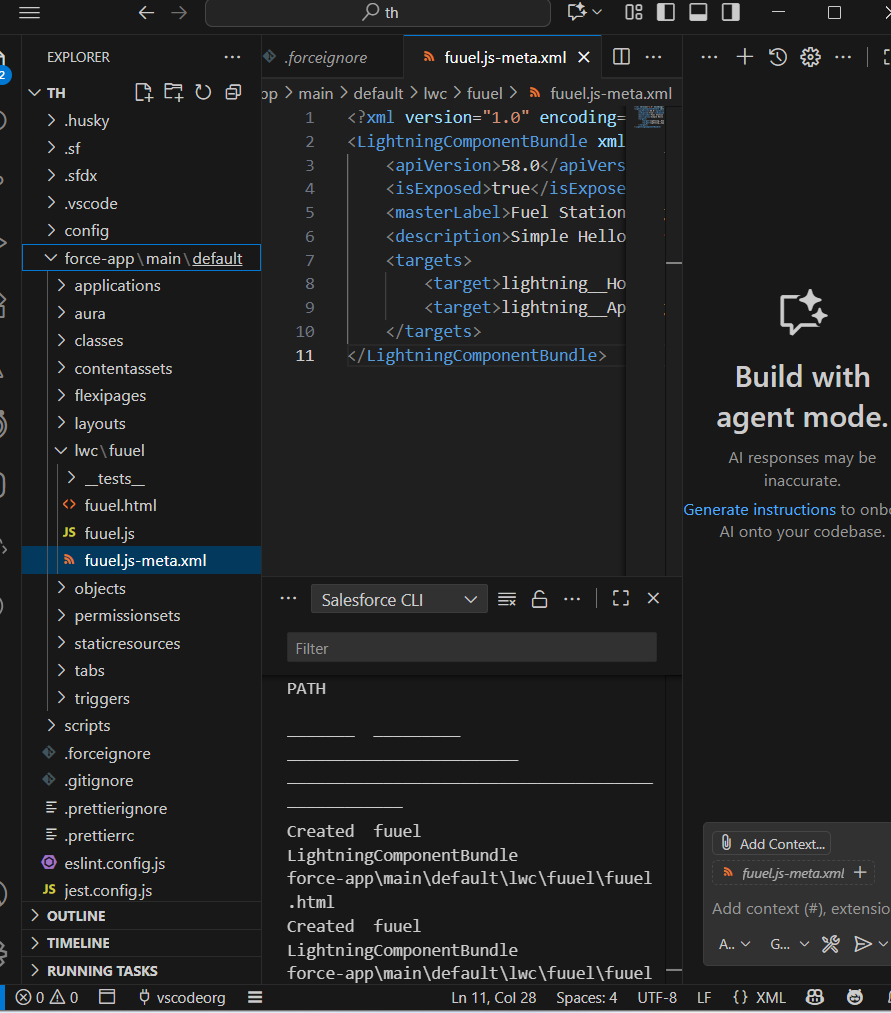
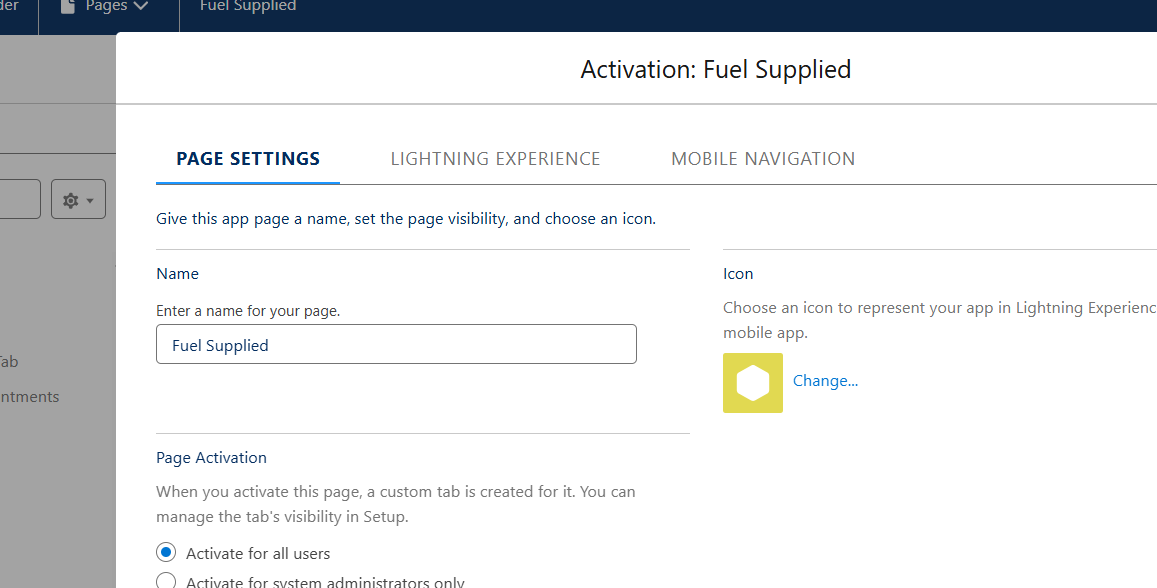
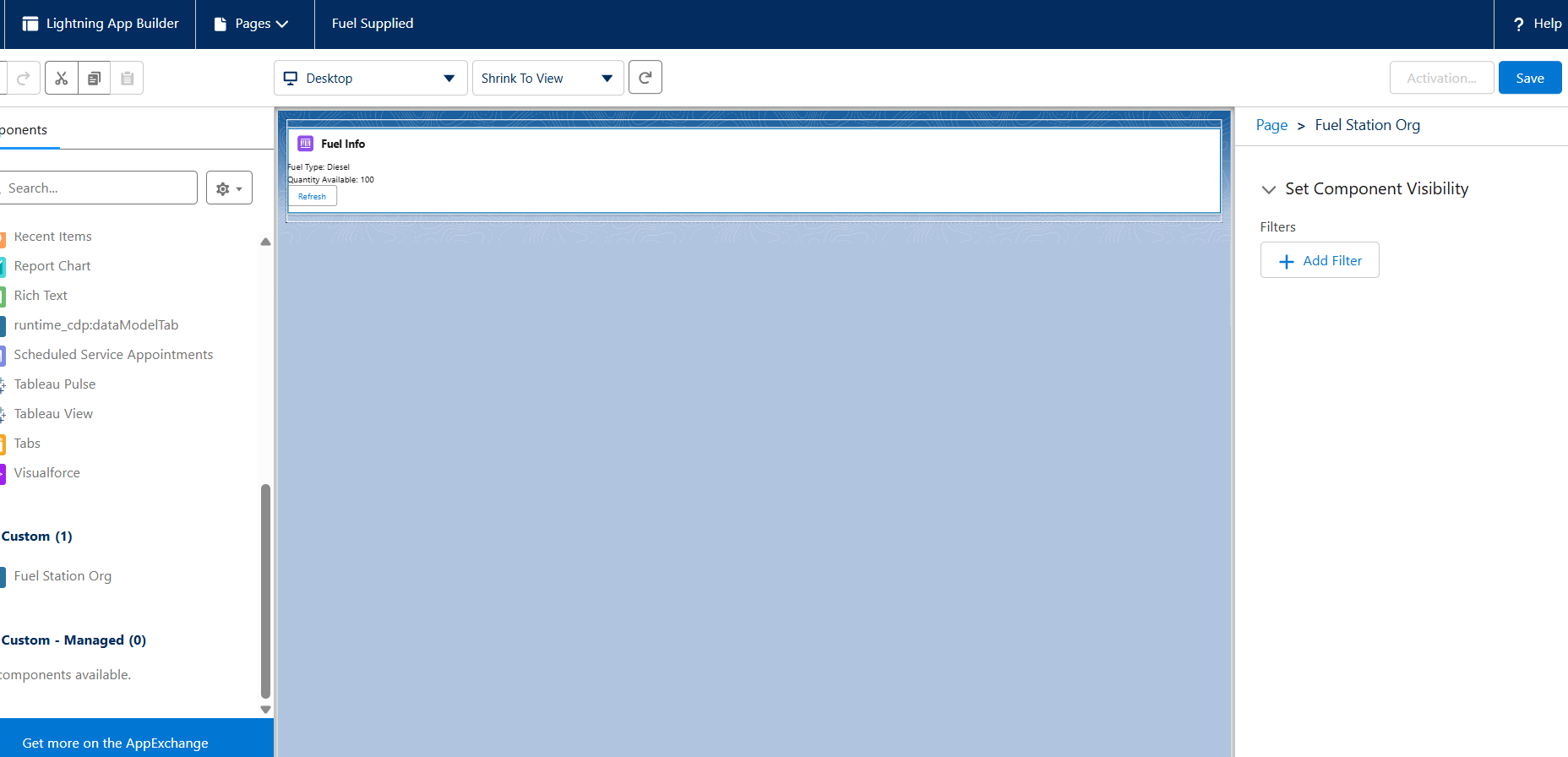
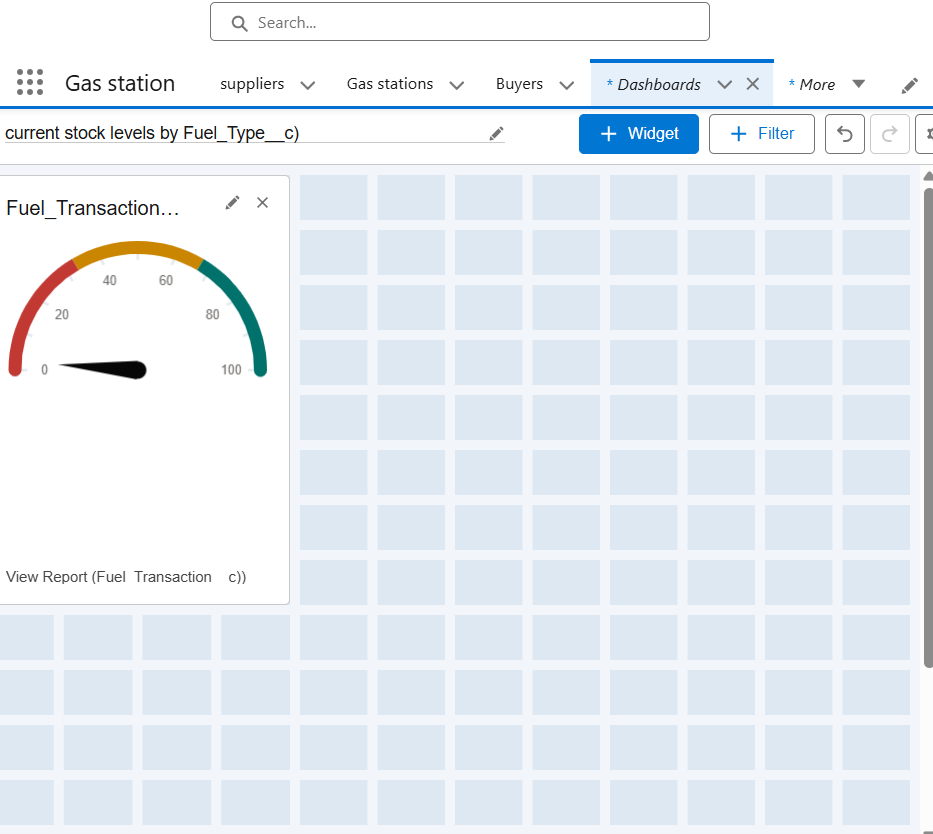
• **Record Pages:** Fuel → list of Booking types of fuel

• **Tabs:** Add Station & Bookings Fuels.

• **Events in LWC:** Child component (search form) → sends event to parent (results).

• **Utility Bar:** Quick “Fuel details” action.

• **Navigation Service:** After booking → navigate to the Booking record page.



✅ Outcome: Outcome: A modern and intuitive interface that enhances user adoption.

# Phase 7 – Integration & External Access

Objective: Connect Salesforce with external systems for real-time data access.

• **Named Credentials:** Secure supplier API logins.

• **External Services:** Connect with external insurance verification.

• **REST Integration:** Fetch real-time fuel prices from external systems.

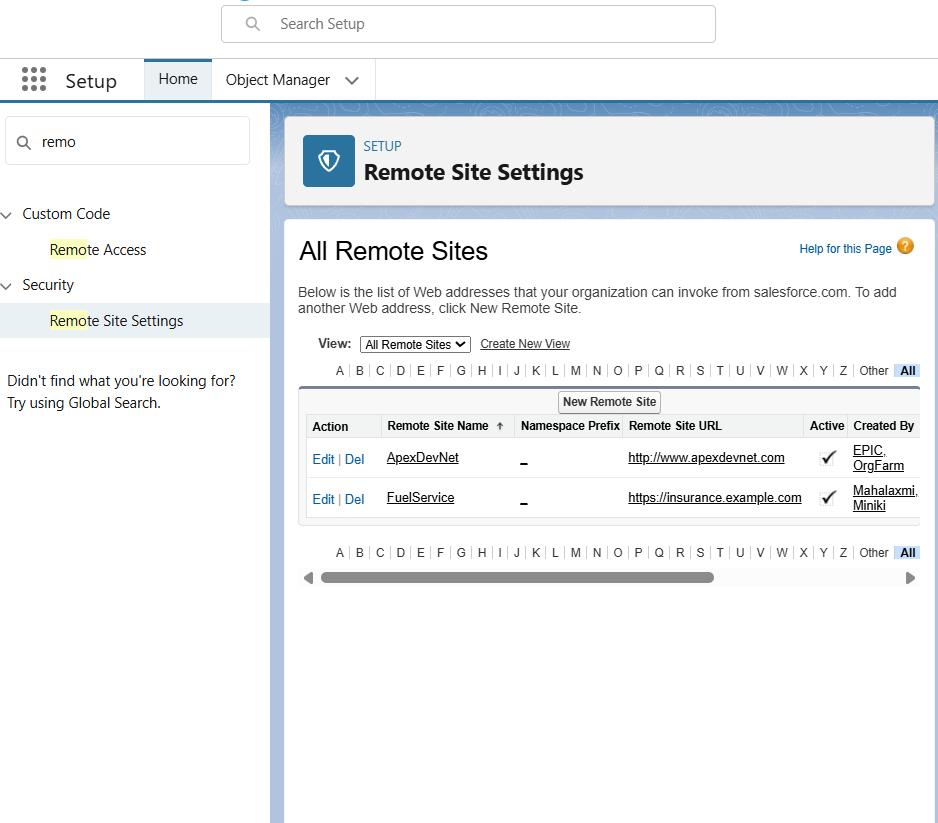
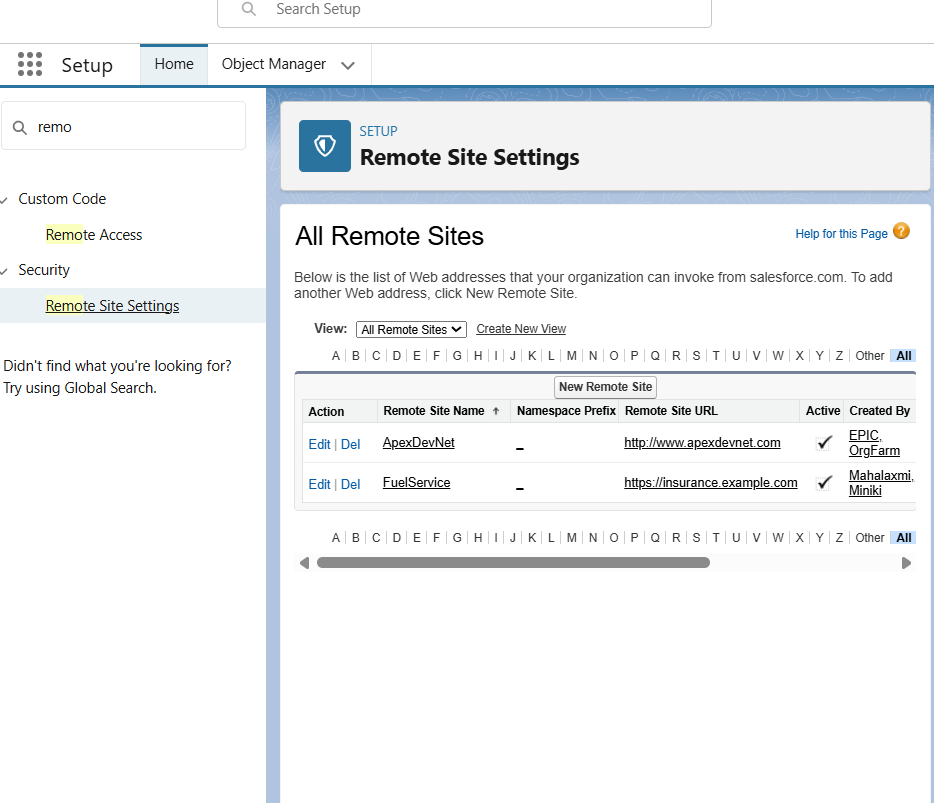
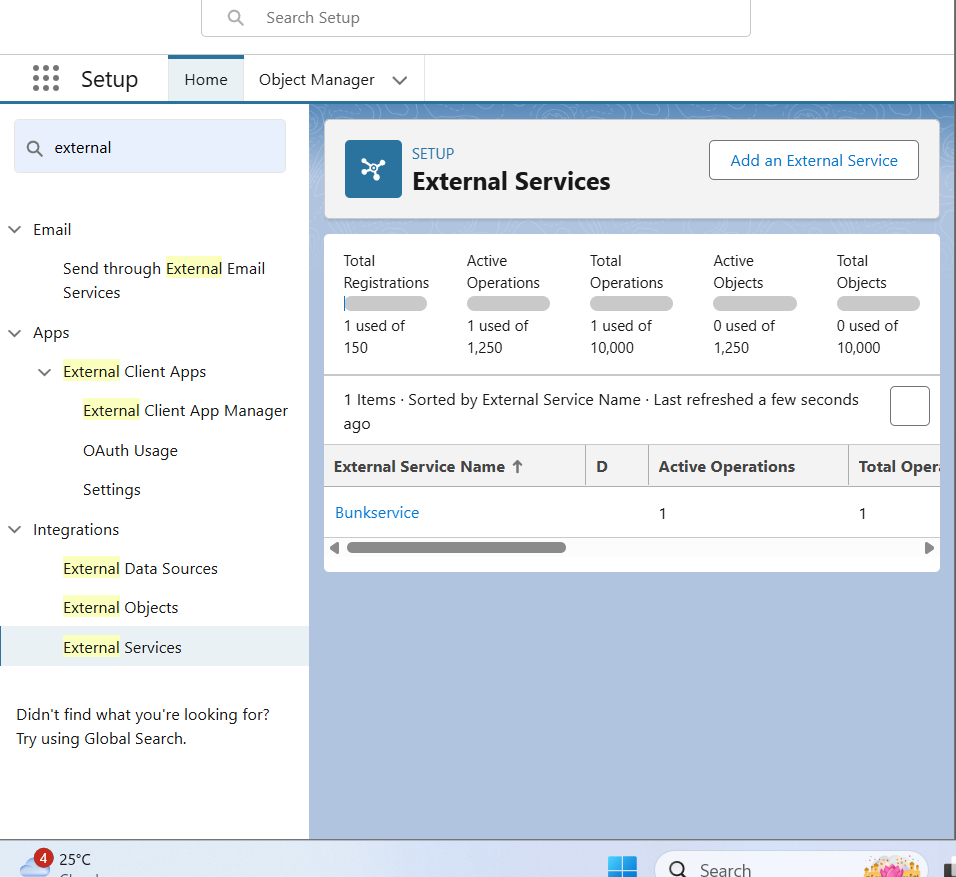
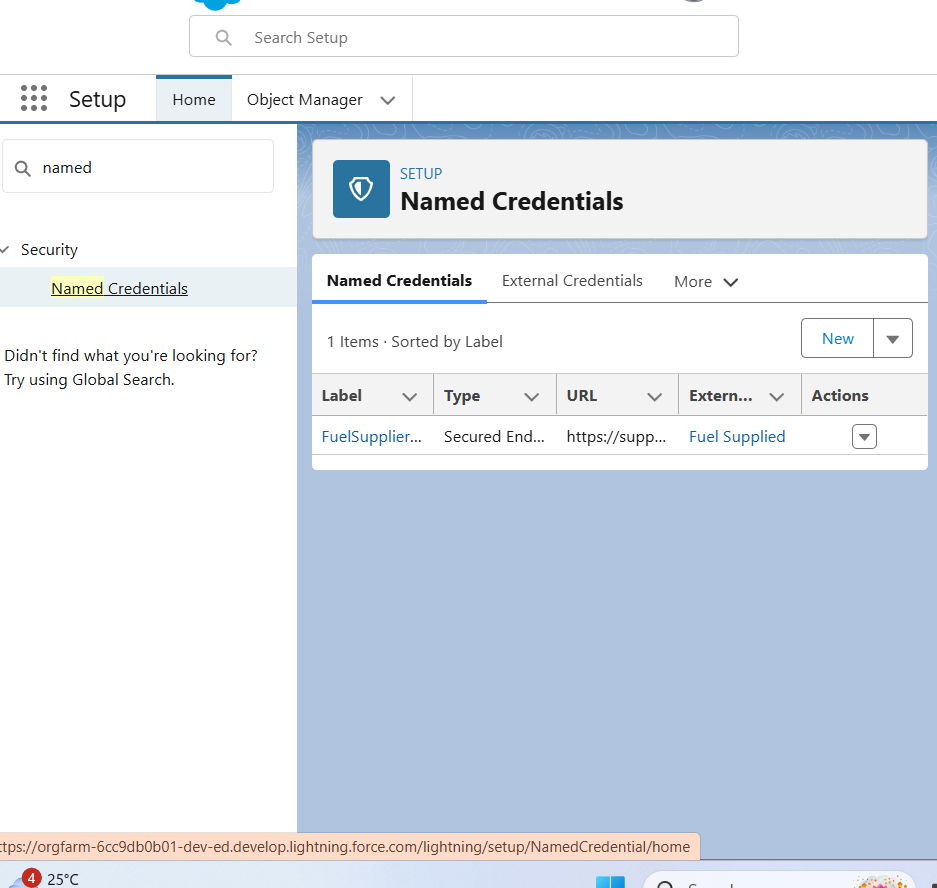
• **Platform Events:** Trigger alerts for low stock levels.

• **OAuth:** Enable secure buyer login via external portals.

• **Remote Site Settings:** Allow callouts to external domains.

• **Callouts:** Triggered when booking was created.

• **API Limits:** Monitor API calls/day.



✅ Outcome: Outcome: Seamless data exchange between Salesforce and external systems.

# Phase 8 – Data Management & Deployment

Objective: Ensure data accuracy and prepare for deployment.

• **Data Import Wizard:** Load supplier and buyer information.

• **Data Loader:** Bulk upload large transaction datasets.

• **Duplicate Rules:** Prevent duplicate records for suppliers and buyers.

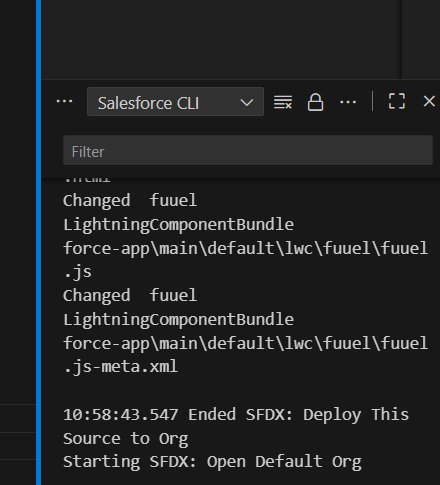
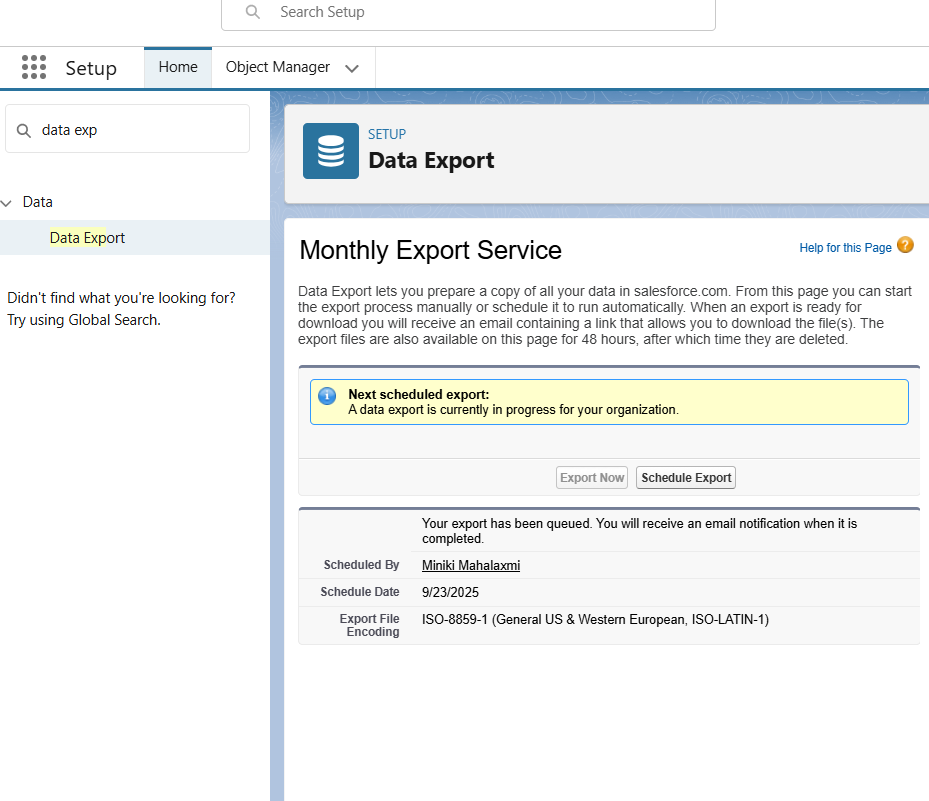
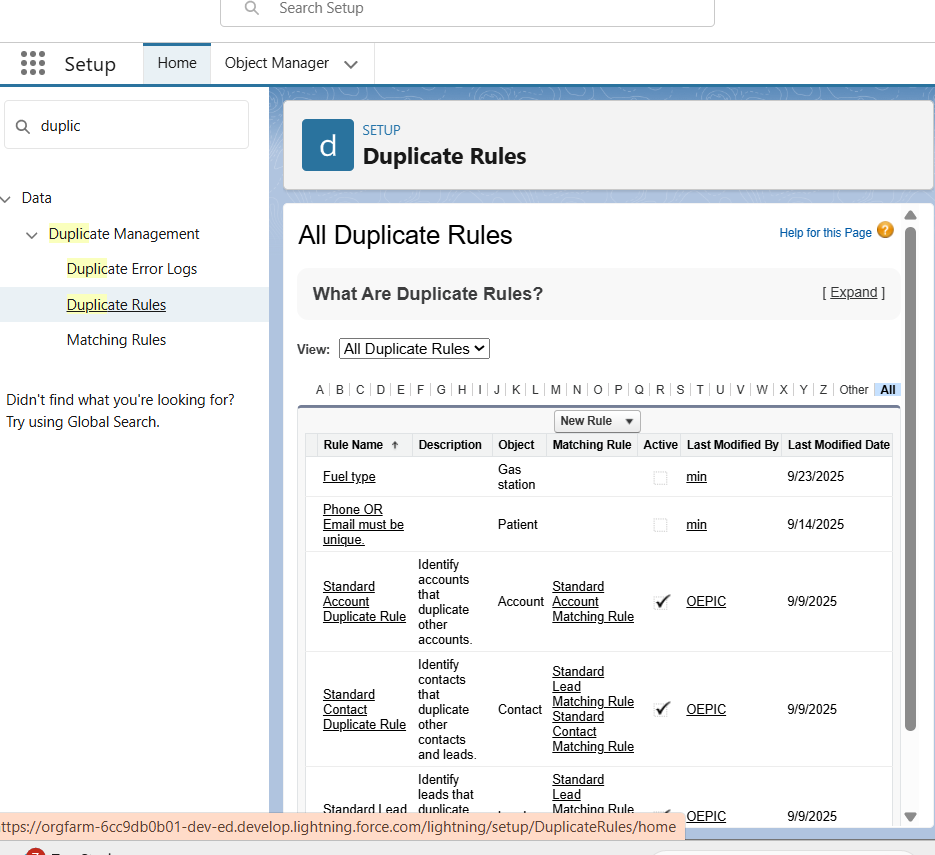
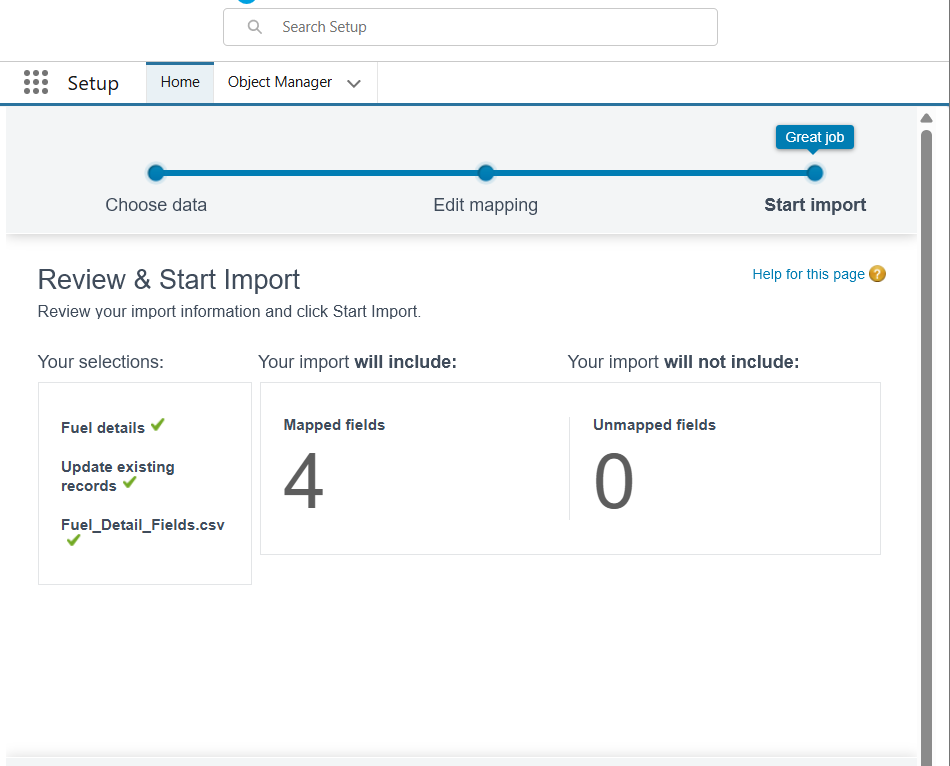
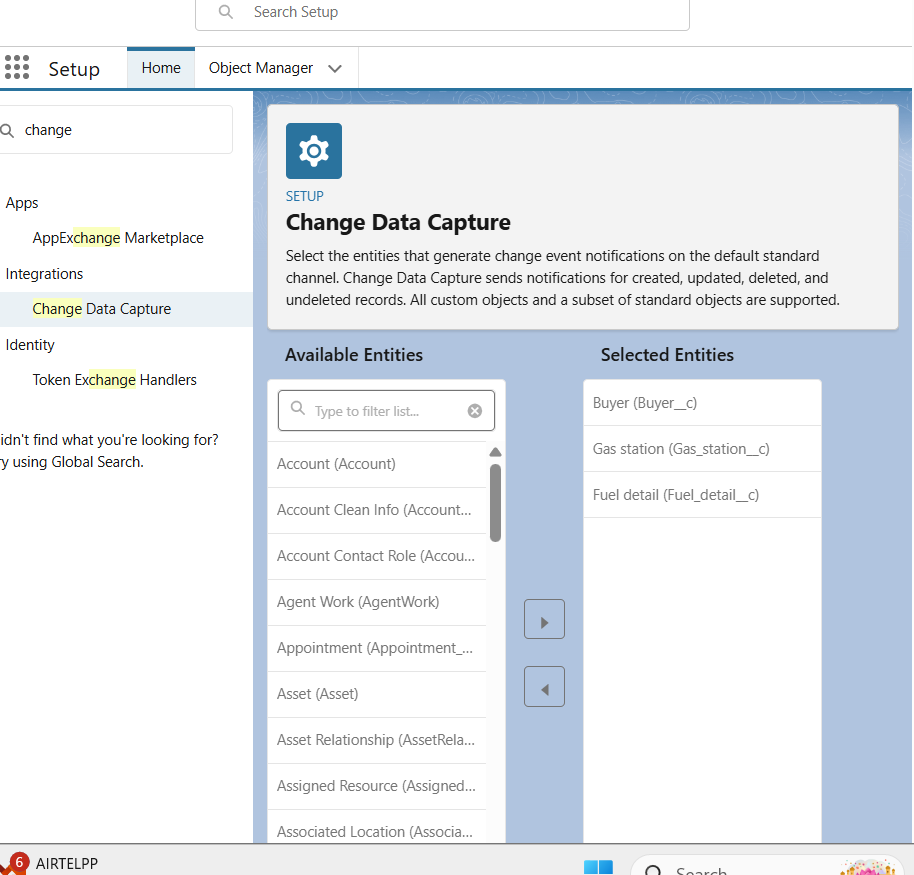
• **Data Backup:** Schedule weekly backups for recovery.

• **Change Sets:** Deploy configurations and customizations between environments.

• **Unmanaged vs Managed Packages:** Managed if you want to publish on AppExchange.

• **ANT Migration Tool:** Command-line deployment.

• **VS Code & SFDX:** Dev-friendly deployments.



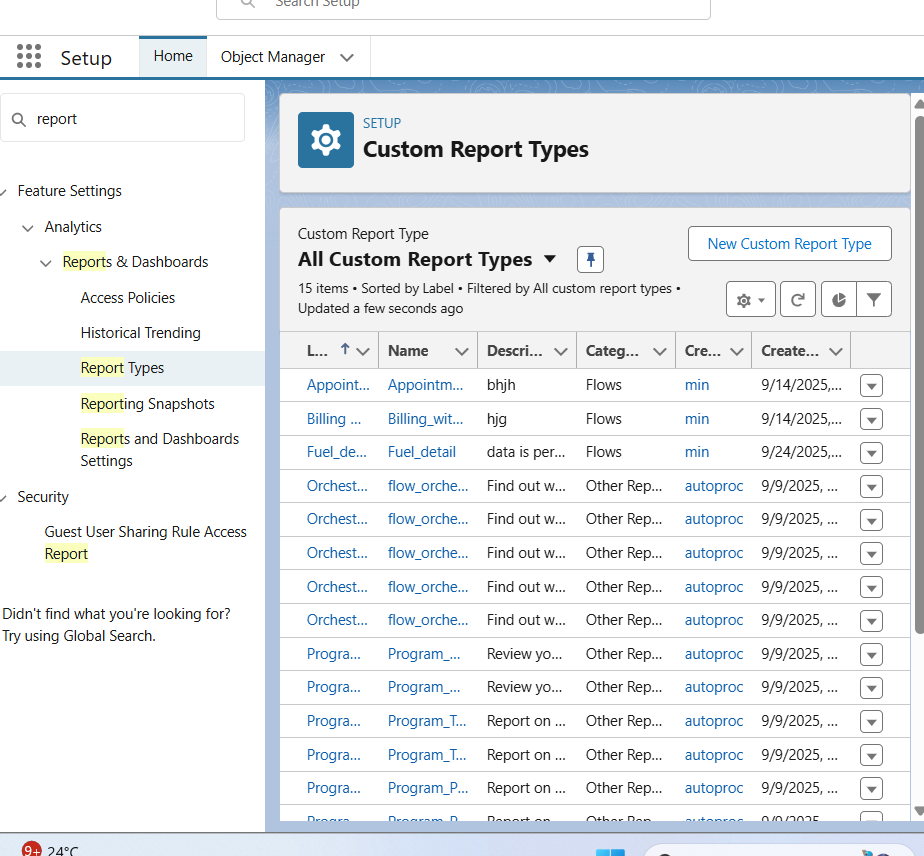
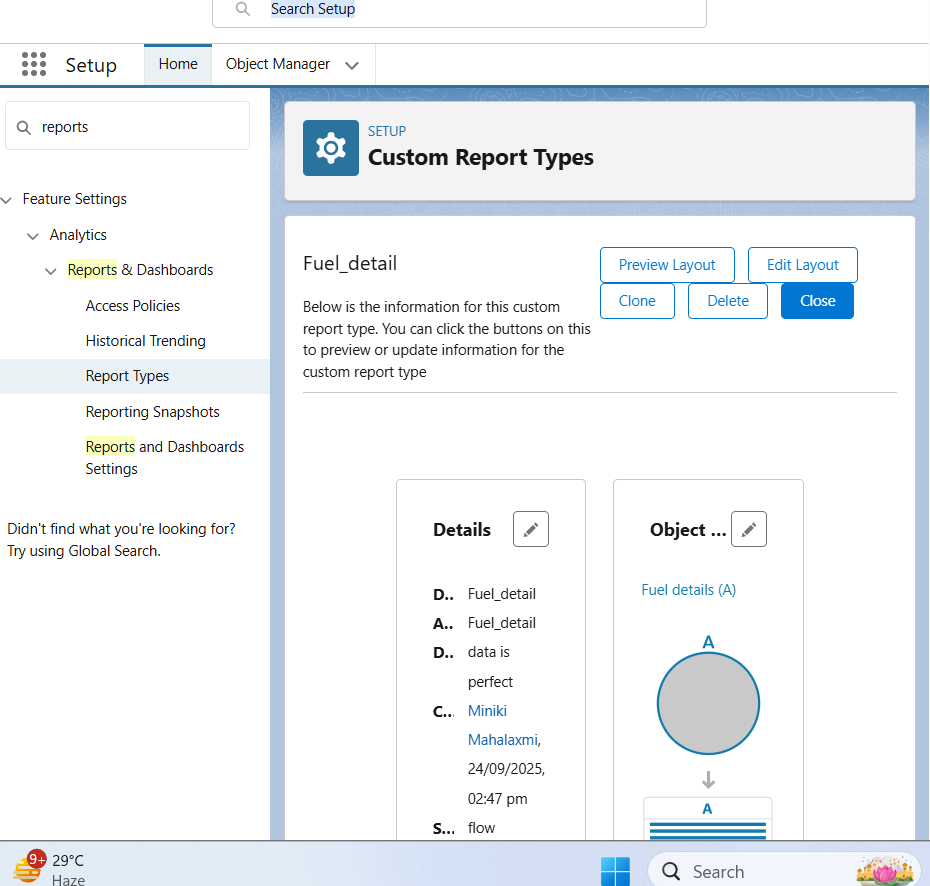
✅ Outcome: Outcome: Reliable, duplicate-free data with a structured deployment strategy.

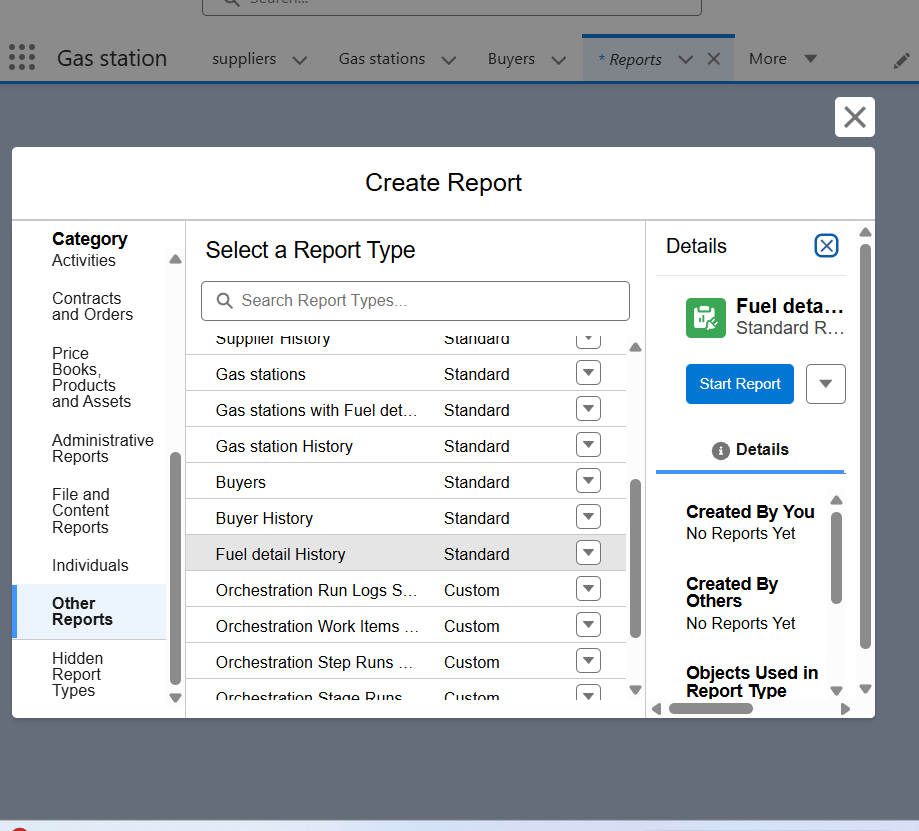
# Phase 9 – Reporting, Dashboards & Security Review

Objective: Provide insights and enforce strong security.

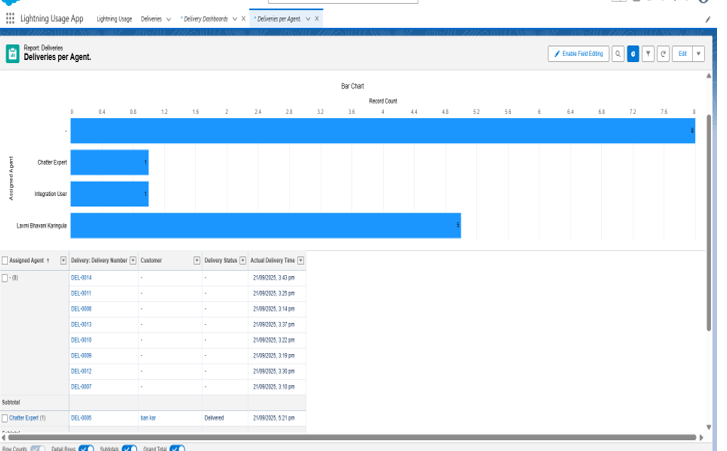
• Reports: Tabular, Summary, Matrix, and Joined reports for sales, supplier performance, and buyer loyalty.

• **Report Types:** Standard and Custom report types for fuel station-specific insights.

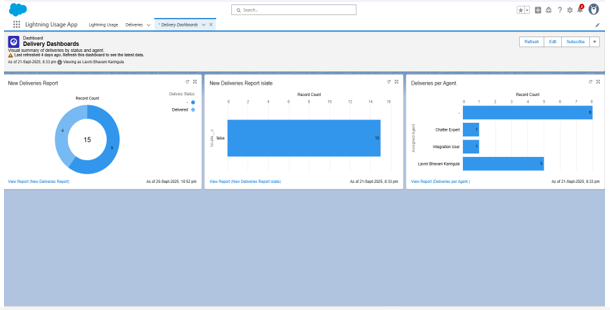




• **Dynamic Dashboards:** Role-based visibility for Admin, Manager, Supplier, Buyer.



• **Dashboards:** Revenue trends, Stock usage, Supplier performance, Buyer loyalty programs.



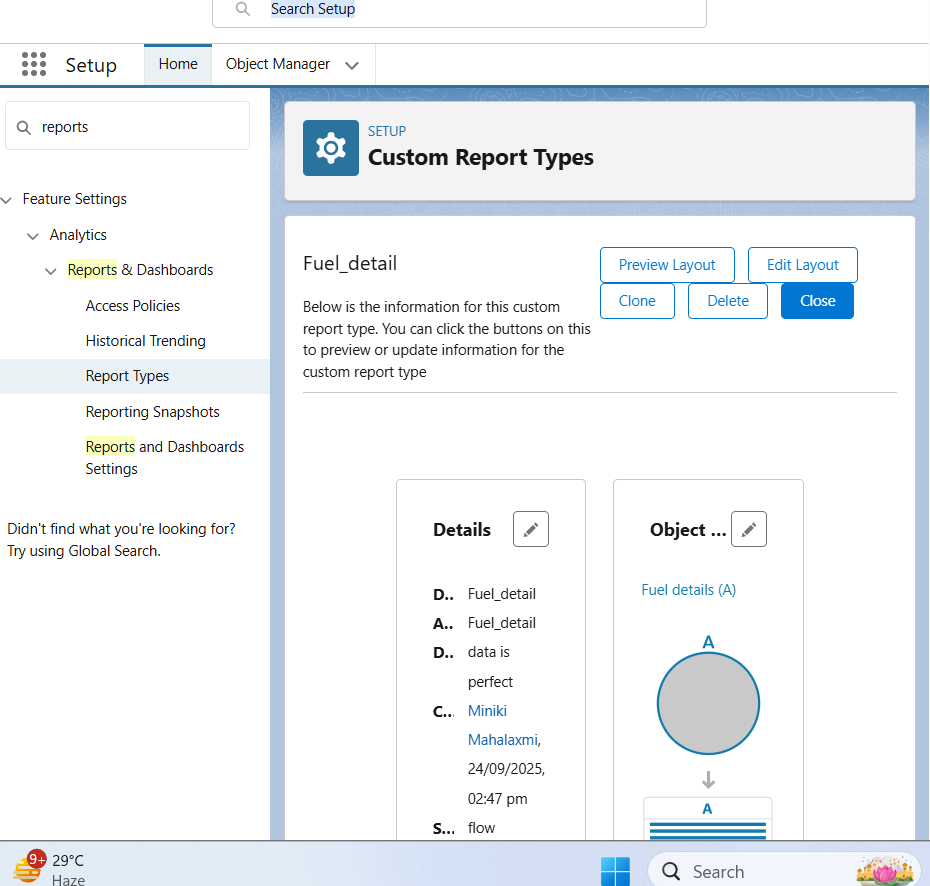
• **Field-Level Security (FLS):** Restrict sensitive data like pricing and customer PII.

• **Sharing Settings:** Configure OWD and Sharing Rules for secure data visibility.

• **Session Settings:** Define session duration, MFA, and timeouts.

• **Login IP Ranges:** Restrict logins to authorized networks.

• **Audit Trail:** Monitor configuration changes and maintain compliance.



✅ Outcome: Outcome: Powerful analytics and dashboards with role-based security and compliance monitoring.

# Phase 10 – Final Presentation & Demo Day

Objective: Showcase the final CRM solution to stakeholders.

• Pitch: Present the problem, solution, and business benefits.

• Demo: Walk through key flows – creating transactions, updating stock, generating reports.

• Documentation: Deliver a comprehensive User and Admin guide.

• Showcase: Publish project details on LinkedIn or professional portfolio.

✅ Outcome: Outcome: A successful project delivery with clear demonstration of value and professional presentation.