Retail Management App

Project Overview:

A Retail Loyalty and Feedback CRM project is developed to help businesses strengthen customer relationships by tracing purchases, fulfilling loyalty, and collecting useful feedback. The CRM allows retailers to manage customer profiles, track purchase history, assign loyalty points, and provide personalized offers to enhance customer retention. It also records customer feedback, which provides discoveries into satisfaction levels, product choices, and areas for service development. Key features involve centralized customer data, loyalty program management, computerized offer tracking, and combined feedback analysis. The business need for this CRM emerges from the growing need to improve customer engagement, establish loyalty, and use data-driven observations to stay competitive in the retail market.

Objective:

The main goal of creating the Retail Loyalty and Feedback CRM is to create a integrated system that enhances customer engagement, drives loyalty sales, and enhances service quality. By focusing customer information such as purchase history, loyalty points, and feedback, the CRM assures better customer management and supports businesses deliver customized offers and rewards that enhance retention. Gathering structured feedback enables retailers to identify trends, solve issues quickly, and upgrade products or services based on customer expectations. This causes streamlined operations, more focused marketing, and statistical decision-making. Ultimately, the CRM grows revenue value by increasing customer satisfaction, increasing loyalty-driven revenue, and offering retailers a strategical advantage in the market.

Phase 1: Problem Understanding & Industry Analysis

1. **Requirement Gathering**
    - Specialized loyalty program (no common discounts).
    - Real-time feedback gathering (surveys, SMS, social media).
    - Predictive churn tracking.
    - Focused campaigns for high-value customers.
2. **Stakeholder Analysis**
    - **Store Managers:** Trace loyalty, validate high-value offers.
    - **Marketing Teams:** Plan & run categorized campaigns.
    - **Customer Support Teams:** Solve negative feedback promptly.
    - **Customers:** Collect points, offers, and share feedback.
3. **Business Process Mapping**
    - Purchase → Points assigned → Feedback collected → Sentiment analysed → Offers generated → Negative feedback escalated → Reports/dashboard updated.
4. **Industry Use Case Analysis**
    - Retail bears from high churn due to common promotions.
    - Competitors target on instant offers; few concentrate on personalized, AI-driven campaigns.
    - Combining loyalty + feedback in one CRM system is a differentiator.
5. **AppExchange Exploration**
    - Existing loyalty apps remain but need real-time feedback integration.

Phase 2: Org Setup & Configuration

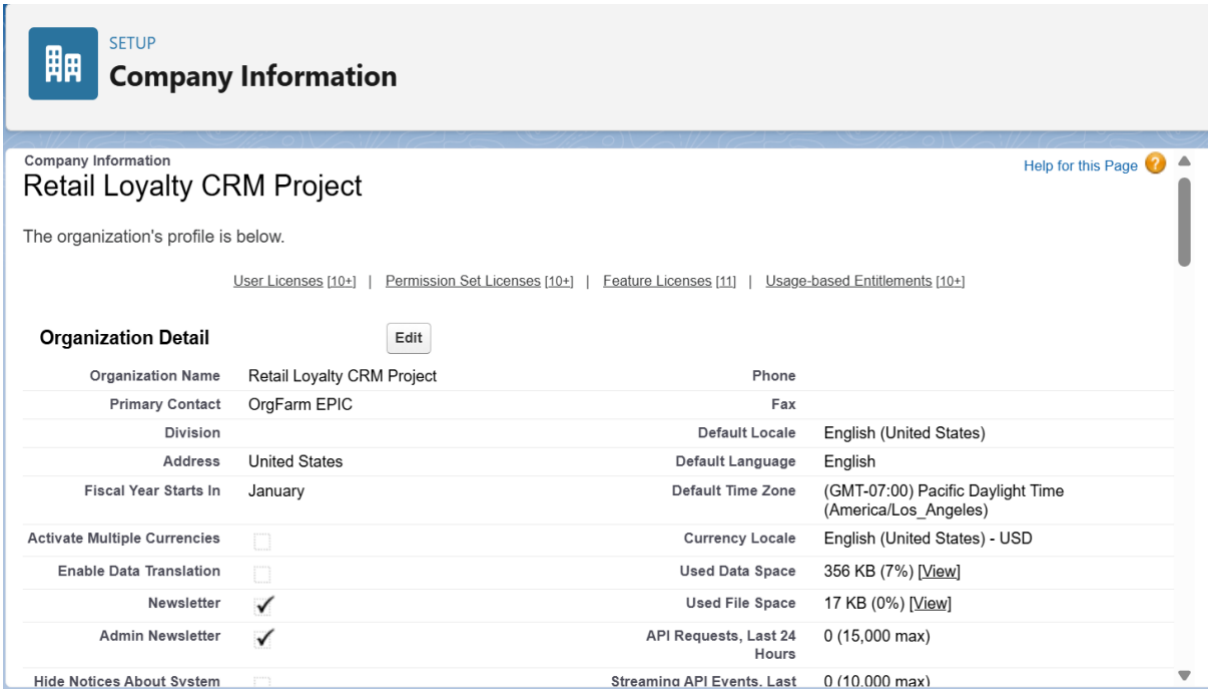
1. **Salesforce Editions**

   ○Use **Developer Edition Developer Org** (free dev org).

2. **Company Profile Setup**

   ○ Go to Company Settings → add company info, local time zone.

   Use Case:
   A national retail company needs to execute a loyalty and feedback CRM over all stores. To assure consistency, the Salesforce Company Profile Setup is planned with the company's fiscal year, currency, time zone, business hours, and default language.

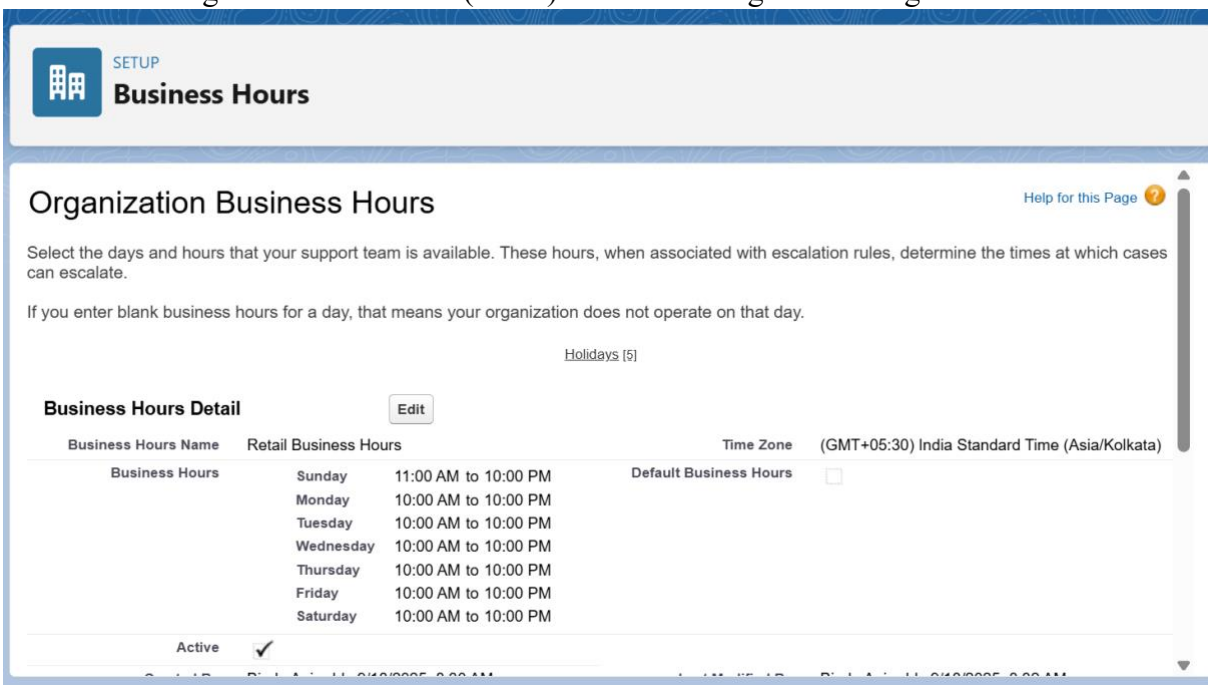


3. **Business Hours & Holidays**

   ○ Define working hours (9am–6pm).

   ○ Add public holidays (no approvals on these days).

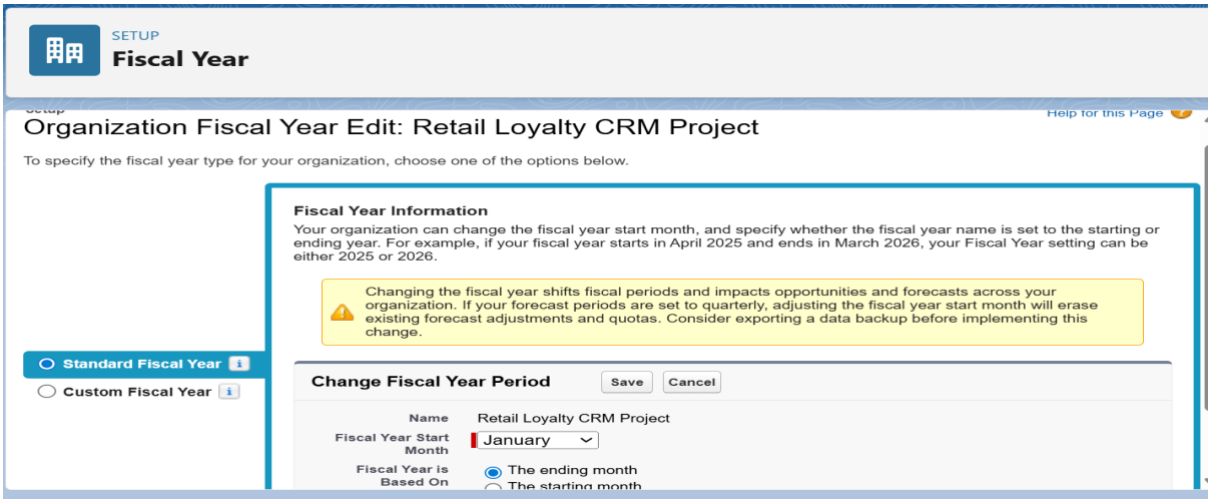
Use Case:
In the retail loyalty and feedback CRM, Business Hours are set to arrange with store operating times (10 AM – 9 PM). Holidays (like Diwali or Christmas) are altered so escalated customer issues don't trigger notifications to service teams on non-working days. This ensures customer problems are managed within agreed service levels (SLAs) without forcing staff during off hours.



4. **Fiscal Year Settings**

○ Standard (Jan–Dec) → good for revenue reporting.

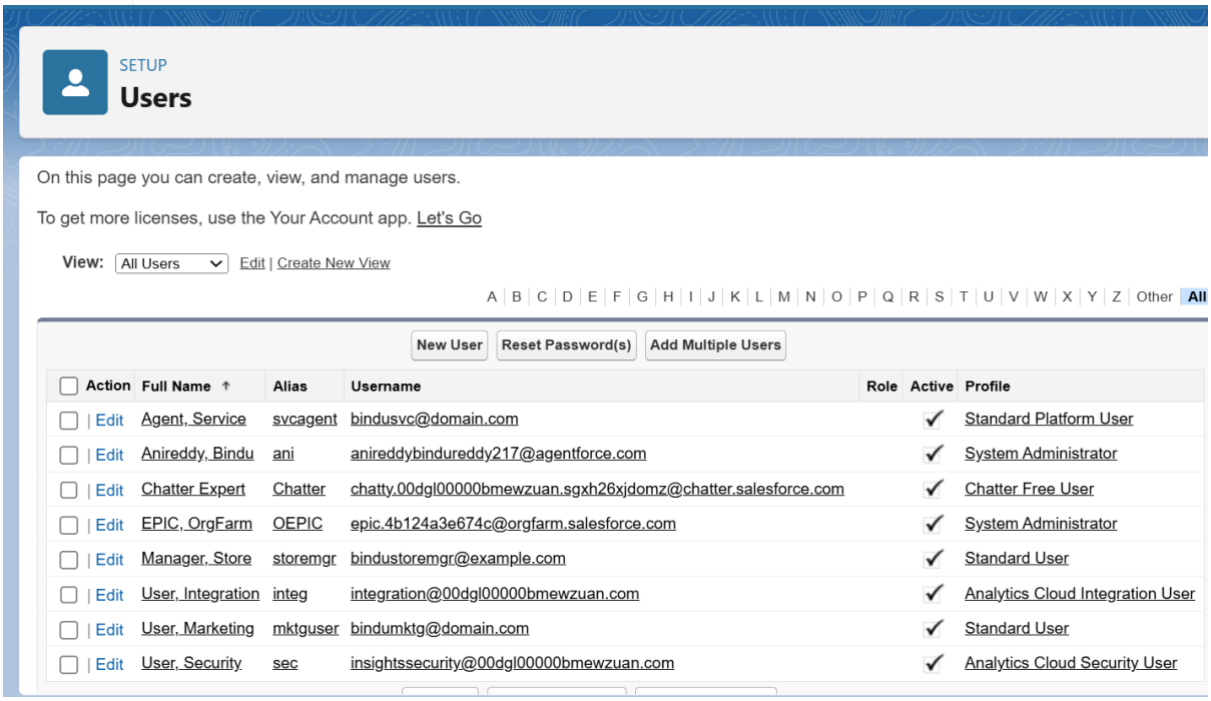
Use Case:
The company runs loyalty point frameworks and redemption reports built on its fiscal year (April–March). Organizing Fiscal Year in Salesforce checks all loyalty analytics, bonus calculations, and financial tracing for customer offers arrange with corporate accounting periods.



5. User Setup & Licenses

○ Create users: Rental Agent, Manager. Assign them Salesforce licenses.

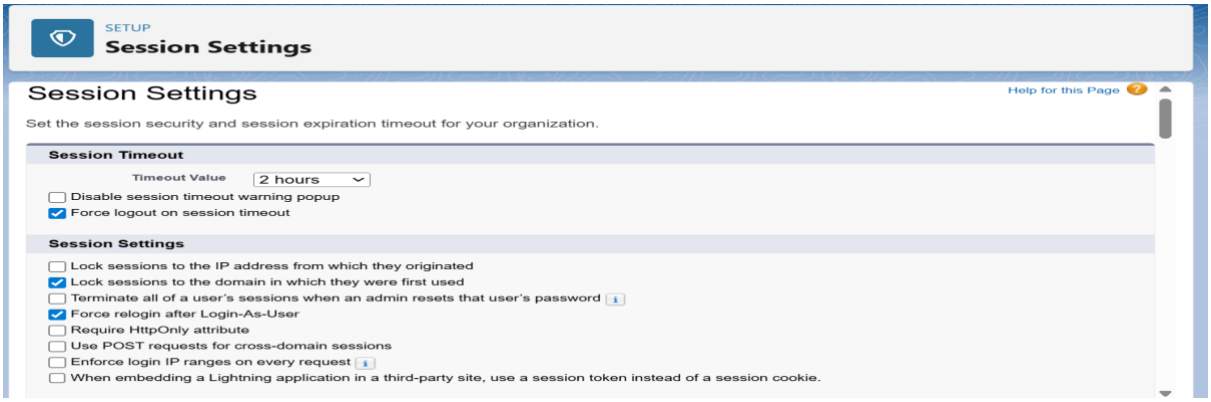
Use Case:
Store Managers, Customer Service Reps, and Regional Admins are organize with relevant Salesforce licenses. Roles and Profiles ensure store managers can trace escalations, while service reps manage feedback cases. This regulated access ensures data security and role-based tasks.

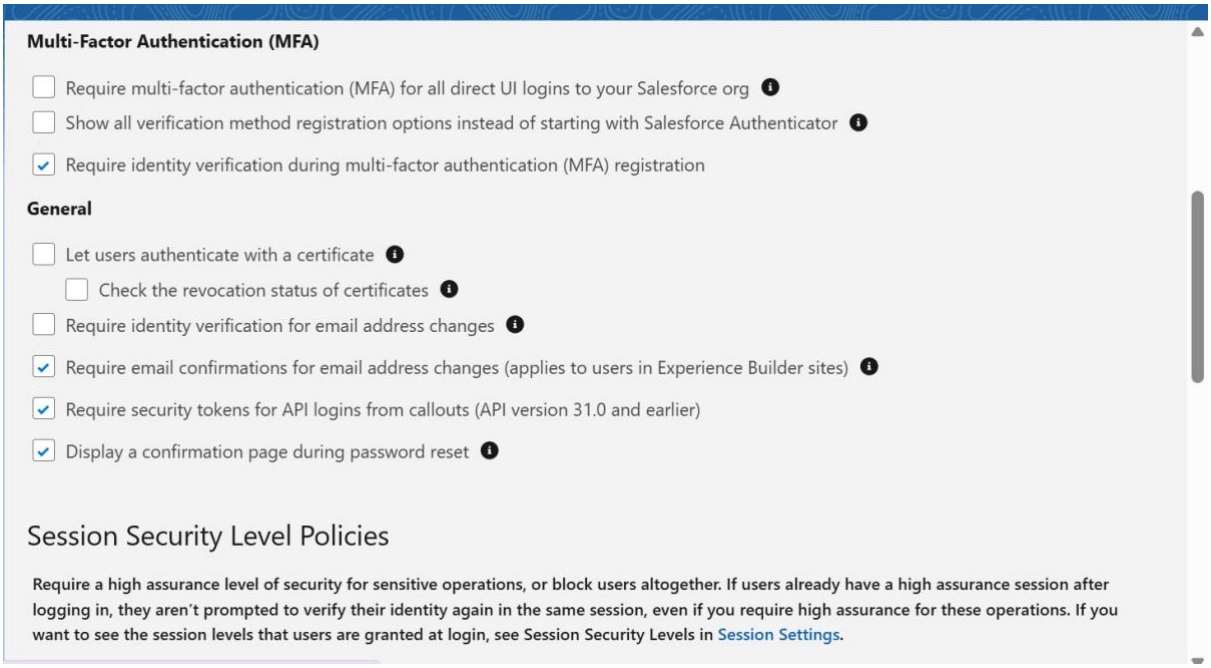


6. **Login Access Policies**

○ Restrict login hours (9am–6pm for agents).

Use Case:
For data security, login IP ranges are regulated to corporate networks, and session timeouts are applied for store devices. Multi-Factor Authentication (MFA) enables that only authorized staff can approach loyalty and feedback data, protecting customer trust.



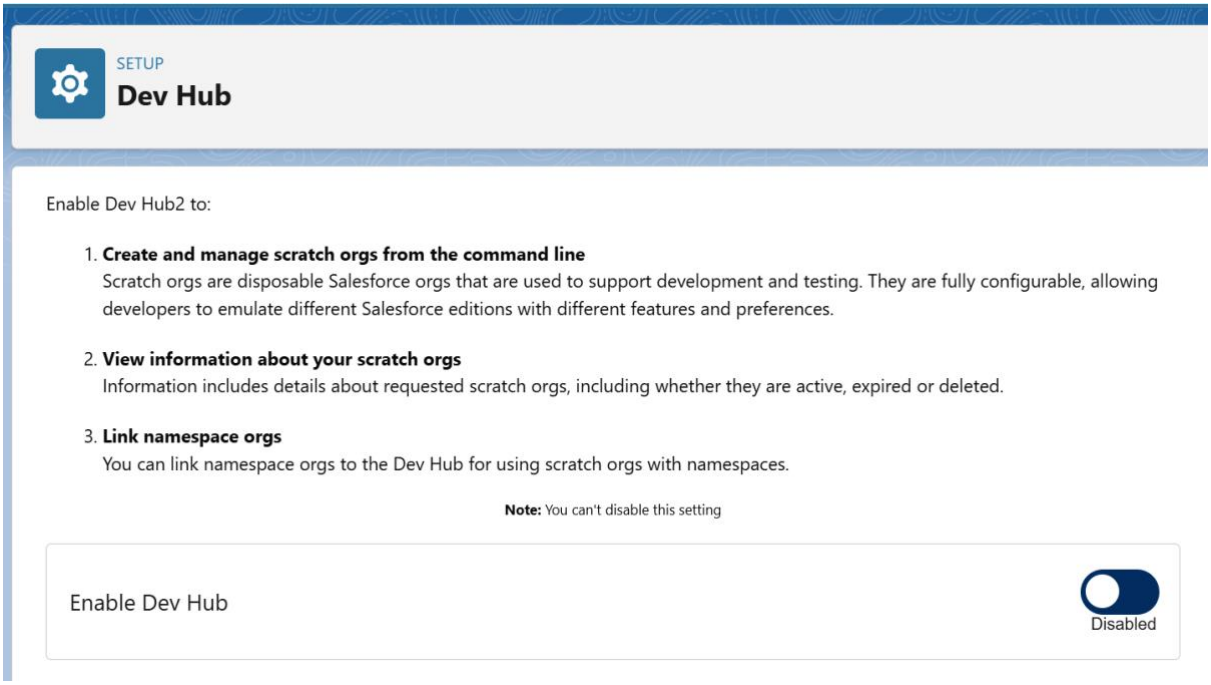


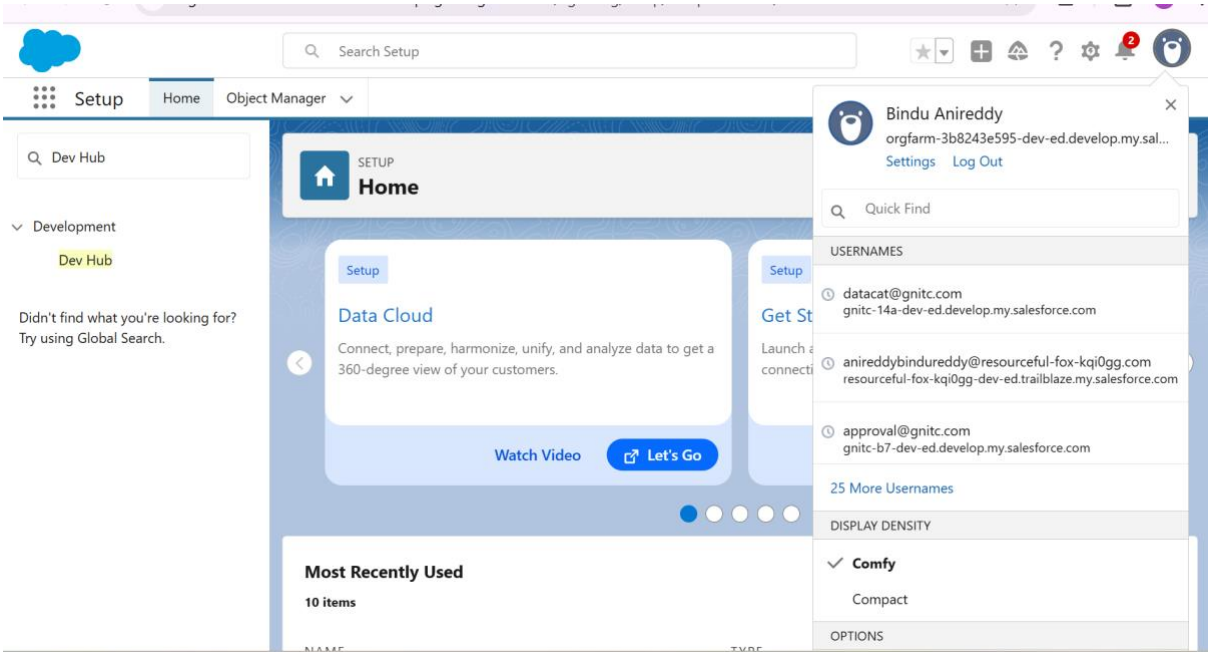
7. **Dev Org Setup**

○This is your sandbox → where you build/test.

Use Case:
A Salesforce Developer Org is constructed to build and test the loyalty CRM aspects (like automated feedback flows and custom notifications) without

impacting production data. This ensures safe prototyping and iterative development.





8. Sandbox Usage

○ If this were a real company, we'd build in Sandbox, then deploy to Production.

Use Case:
A **Full Sandbox** is used to simulate real customer and transaction data for UAT (User Acceptance Testing). A **Developer Sandbox** is operated for individual promoted testing (e.g., complaint escalation flows). This isolation prevents testing errors from affecting live customer data.

9. **Deployment Basics**

○ Deployment is moving config/code from sandbox → production using **Change Sets**.

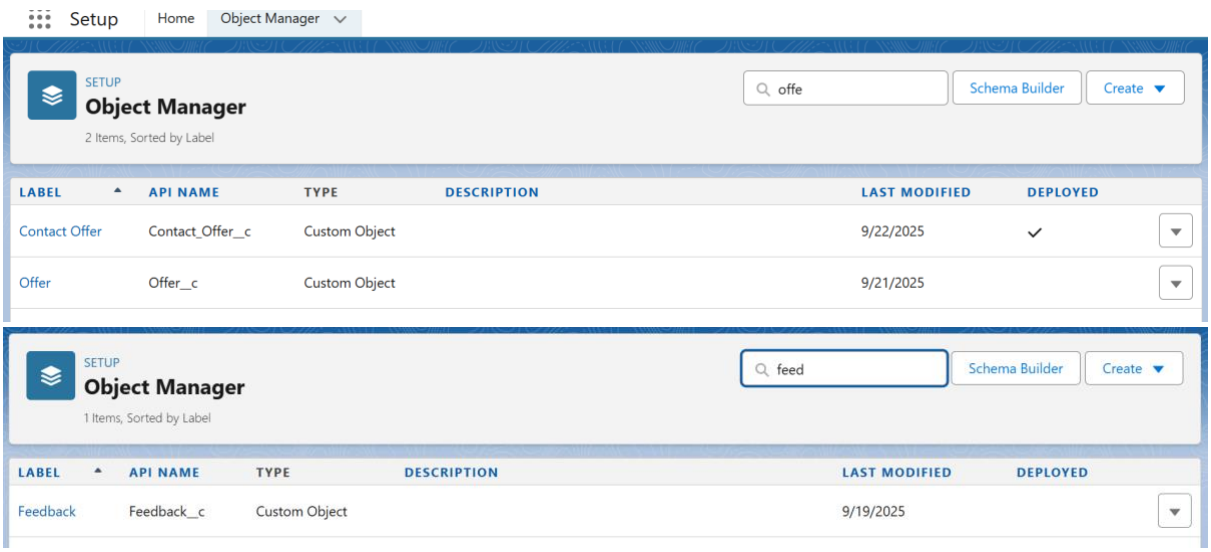
Use case:
New properties, such as feedback automation and loyalty bonus calculation, are first validated in Sandboxes. Change Sets are then deployed to deploy these aspects into Production. This organized deployment assures stable, error-free rollout of CRM operation across all retail stores.

Phase 3: Data Modeling & Relationships

1.Standard & Custom Objects

- Standard objects are Salesforce's pre-configured objects like Account, Contact, Case, or Opportunity. Custom objects are built to store data specific to your business needs that standard objects cannot handle.

- **Use Case:** In a retail loyalty CRM, standard objects like Account and Contact can store and customer information. Custom objects like Feedback, Loyalty_Points, or Offer can trace customer interactions, loyalty rewards, and promotional offers.

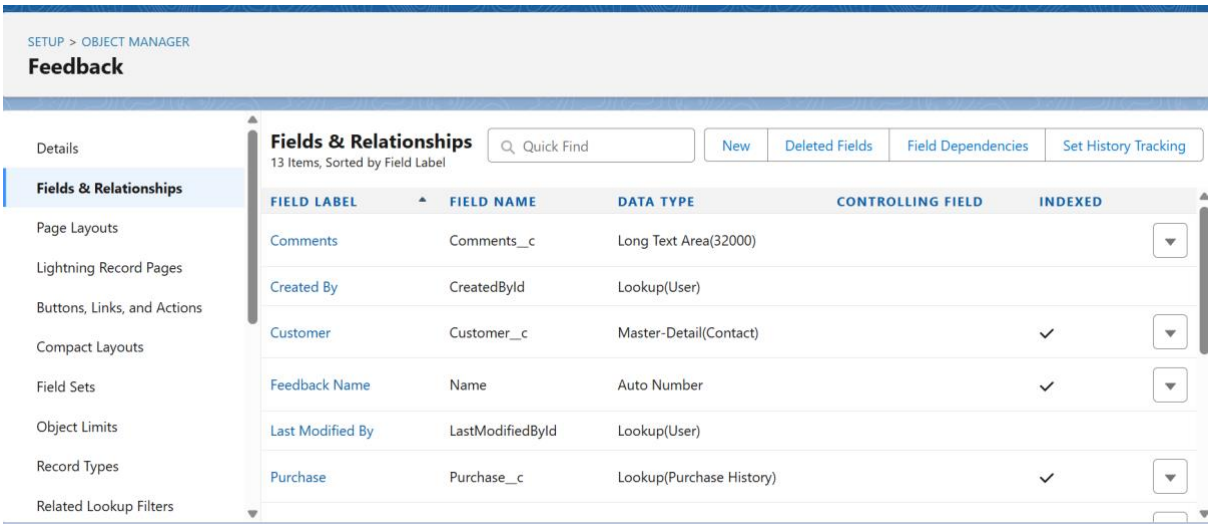


LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Purchase History	Purchase_History__c	Custom Object		9/19/2025	

2. Fields

- Fields are the individual data points retained on objects. Fields can be of types like text, number, picklist, checkbox, or formula.

- **Use Case:** On the Feedback object, you might have fields like Rating (1–5), Comments (text area), and Feedback_Date. These fields enable detailed monitoring of customer satisfaction per store visit.



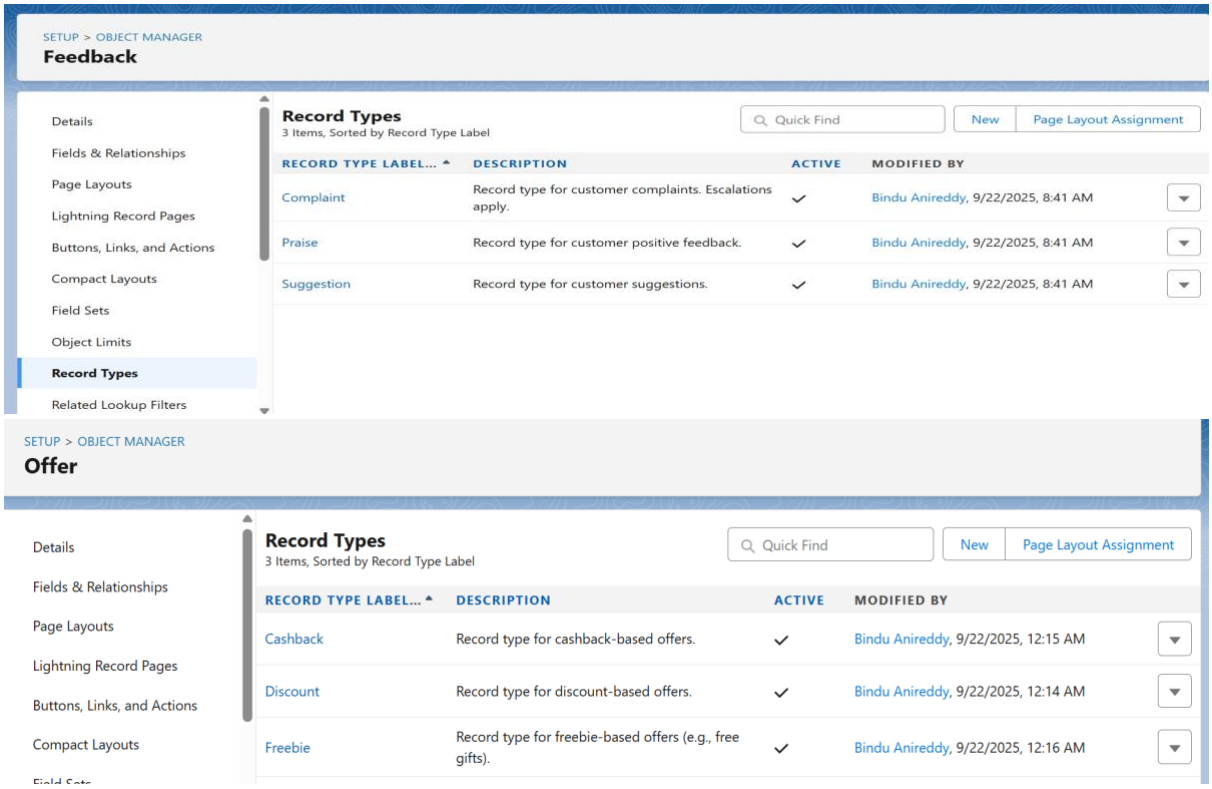
3. Record Types

- Record Types allows the same object to have various business processes, picklist values, and page layouts depending on the type of record.

- **Use Case:**

- Feedback Object: You have three record types:
- Complaint – for negative customer feedback requiring escalation.
- Suggestion – for ideas or improvement requests from customers.
- Positive Note – to track positive feedback or compliments.
  These record types allow the Feedback object to display different fields or picklist values depending on the type, making it easier for staff to categorize and act on feedback.
- Offers Object: You have three record types:
- Discount – e.g., percentage-based discounts.
- Cashback – rewards returned to customers after purchase.

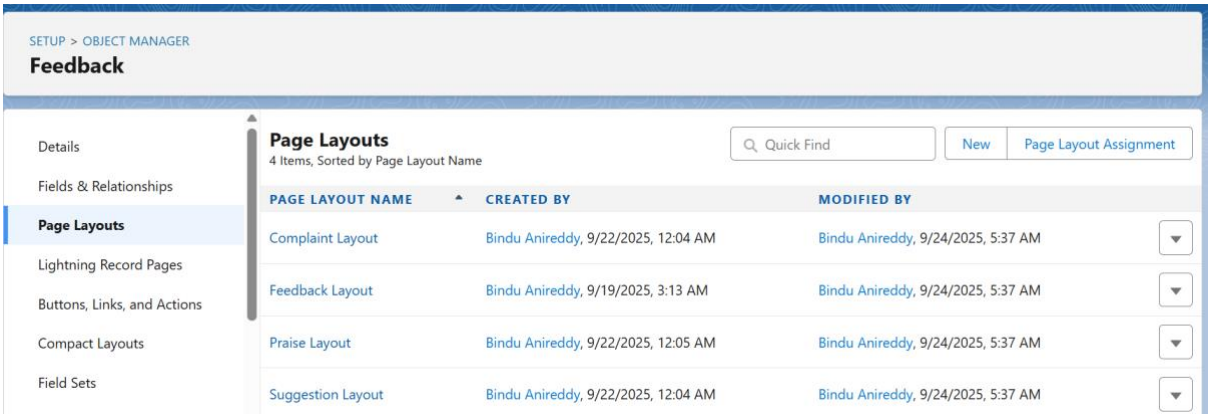
- Freebie – complimentary items or promotional gifts.
  Record Types here enable configuring different fields (like discount %, cashback amount, or free item SKU) and page layouts for each offer type.



4. Page Layouts

- Page Layouts explain how fields, related lists, and sections appear on a record page. They manage the user experience for creating, viewing, or editing records.

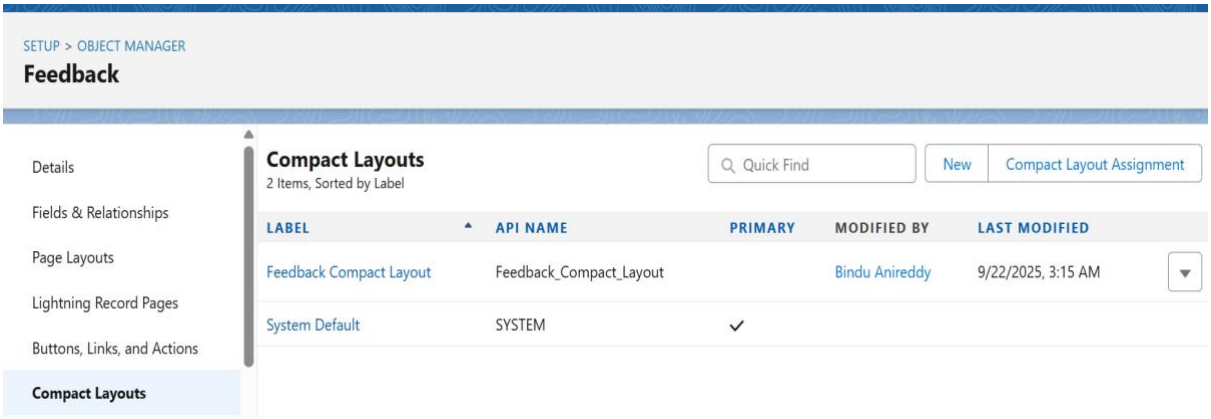
- **Use Case:** On the Customer object, a page layout can focus on loyalty points and feedback history for retail staff, while maintaining administrative details like account creation info in a separate section.



5. Compact Layouts

- Compact Layouts regulate which fields are seen in the highlights panel of a record (e.g., on mobile or Lightning Experience).

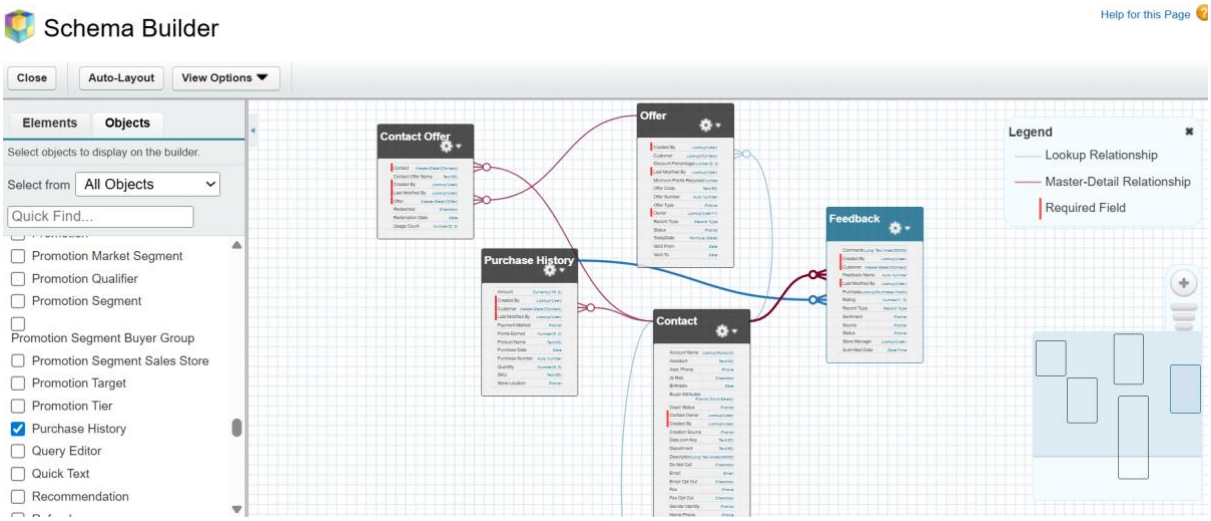
- **Use Case:** For a customer record, a compact layout can show the Name, Loyalty Tier, and Last Purchase Date at a glance, helping store staff instantly understand the customer profile.



6. Schema Builder

- Schema Builder is a visual tool in Salesforce to outlook and govern objects, fields, and relationships in a drag-and-drop interface.

- **Use Case:** In your retail CRM, you can use Schema Builder to quickly visualize how Customer, Feedback, Offer, and Loyalty_Points objects are connected, making it easier to plan relationships.

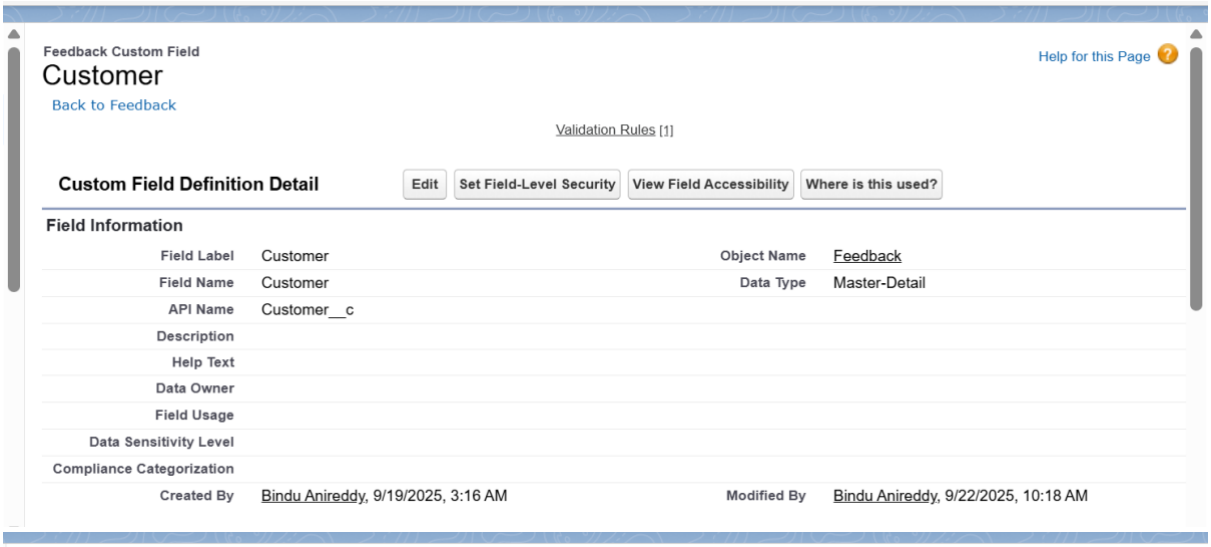


7. Lookup vs Master-Detail vs Hierarchical

- **Lookup Relationship:** Loose connection between two objects; child can exist without parent.
- **Master-Detail Relationship:** Strong connection; child's existence depends on parent, and parent manages sharing & security.
- **Hierarchical Relationship:** Special lookup on the User object to define management chains.

- **Use Case:**

- Customer → Feedback can be a **Lookup** (feedback can exist independently).
- Customer → Loyalty_Points can be **Master-Detail** (points are meaningless without a customer).
- User → Manager can use **Hierarchical** to define store employee reporting structure.



Feedback Custom Field

Customer

Back to Feedback

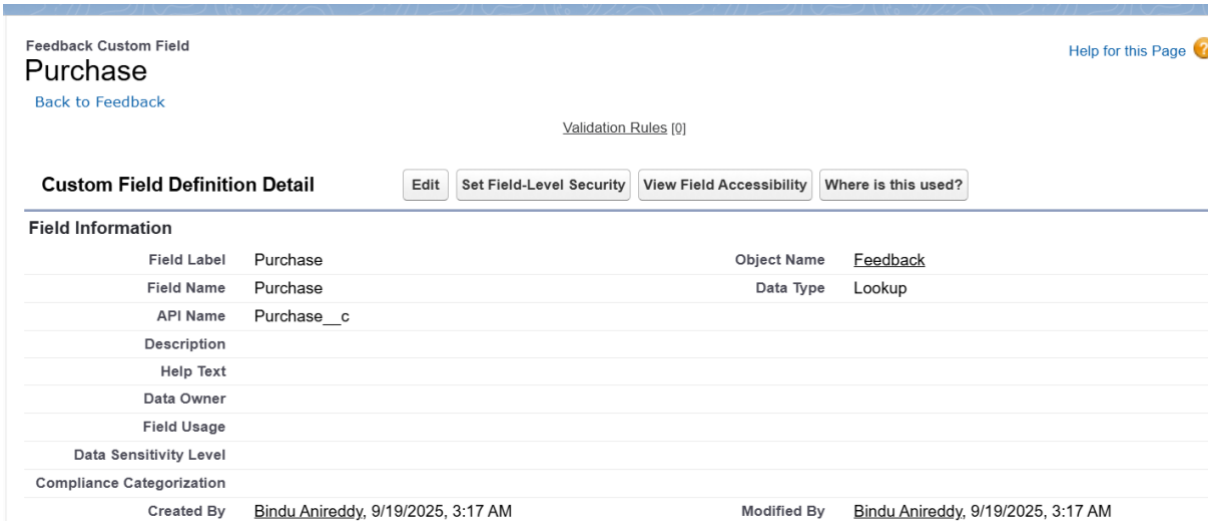
Help for this Page

Validation Rules [1]

Custom Field Definition Detail Edit | Set Field-Level Security | View Field Accessibility | Where is this used?

Field Information

Field Label	Customer	Object Name	Feedback
Field Name	Customer	Data Type	Master-Detail
API Name	Customer__c		
Description			
Help Text			
Data Owner			
Field Usage			
Data Sensitivity Level			
Compliance Categorization			
Created By	Bindu Anireddy, 9/19/2025, 3:16 AM	Modified By	Bindu Anireddy, 9/22/2025, 10:18 AM



Feedback Custom Field

Purchase

Back to Feedback

Help for this Page

Validation Rules [0]

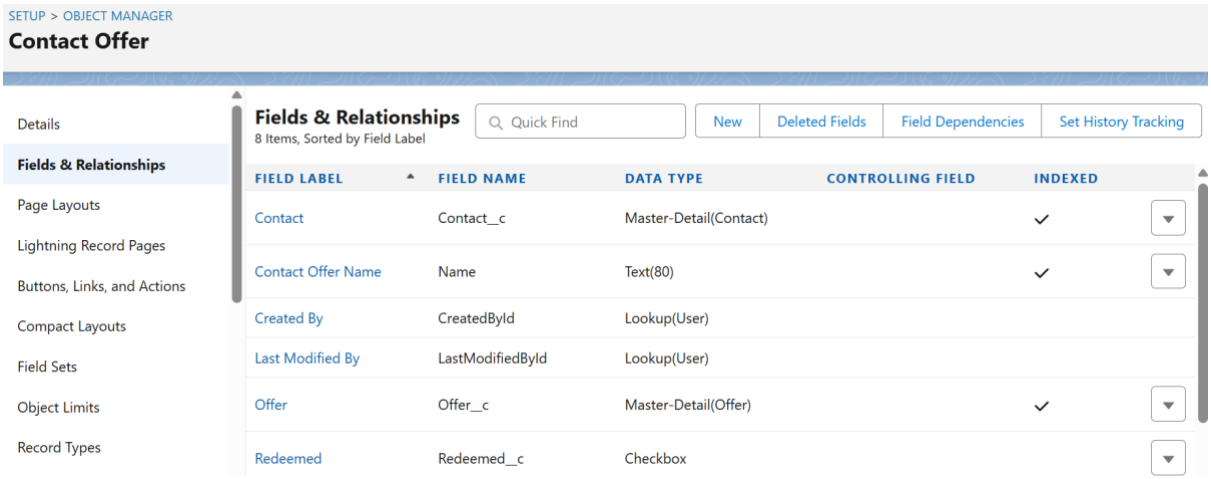
Custom Field Definition Detail Edit | Set Field-Level Security | View Field Accessibility | Where is this used?

Field Information

Field Label	Purchase	Object Name	Feedback
Field Name	Purchase	Data Type	Lookup
API Name	Purchase__c		
Description			
Help Text			
Data Owner			
Field Usage			
Data Sensitivity Level			
Compliance Categorization			
Created By	Bindu Anireddy, 9/19/2025, 3:17 AM	Modified By	Bindu Anireddy, 9/19/2025, 3:17 AM

8. Junction Objects

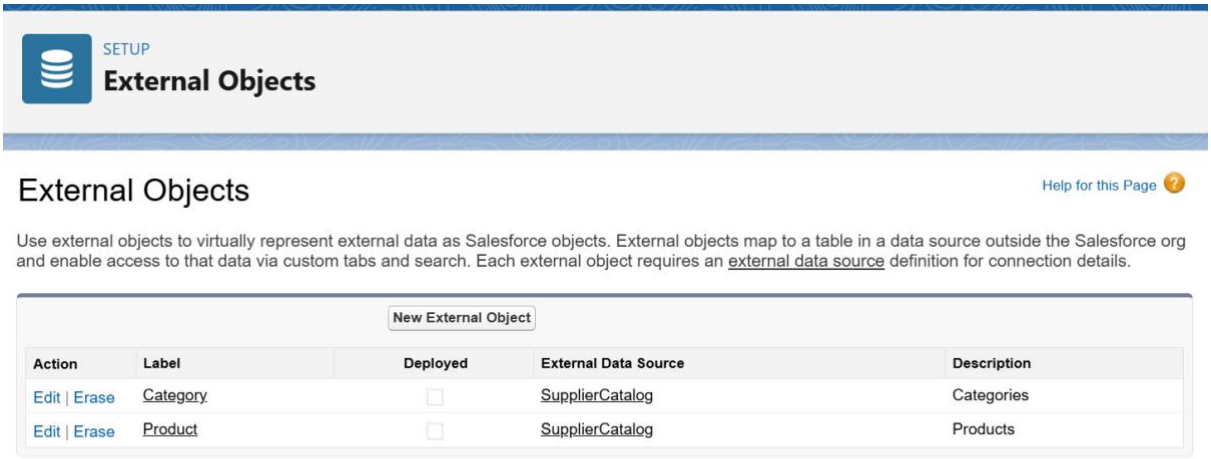
- A junction object enables many-to-many relationships between two objects. It includes two master-detail relationships.
- **Use Case:** For a promotion program, Customer ↔ Offer can be many-to-many. The junction object Customer_Offer monitors which customers received which offers.



9. External Objects

- External objects allow Salesforce to access data stored outside of Salesforce in real-time (via OData or other external sources).

- **Use Case:** If inventory is managed in an external ERP system, an external object Store_Inventory can fetch in stock levels in real-time without storing the data in Salesforce.

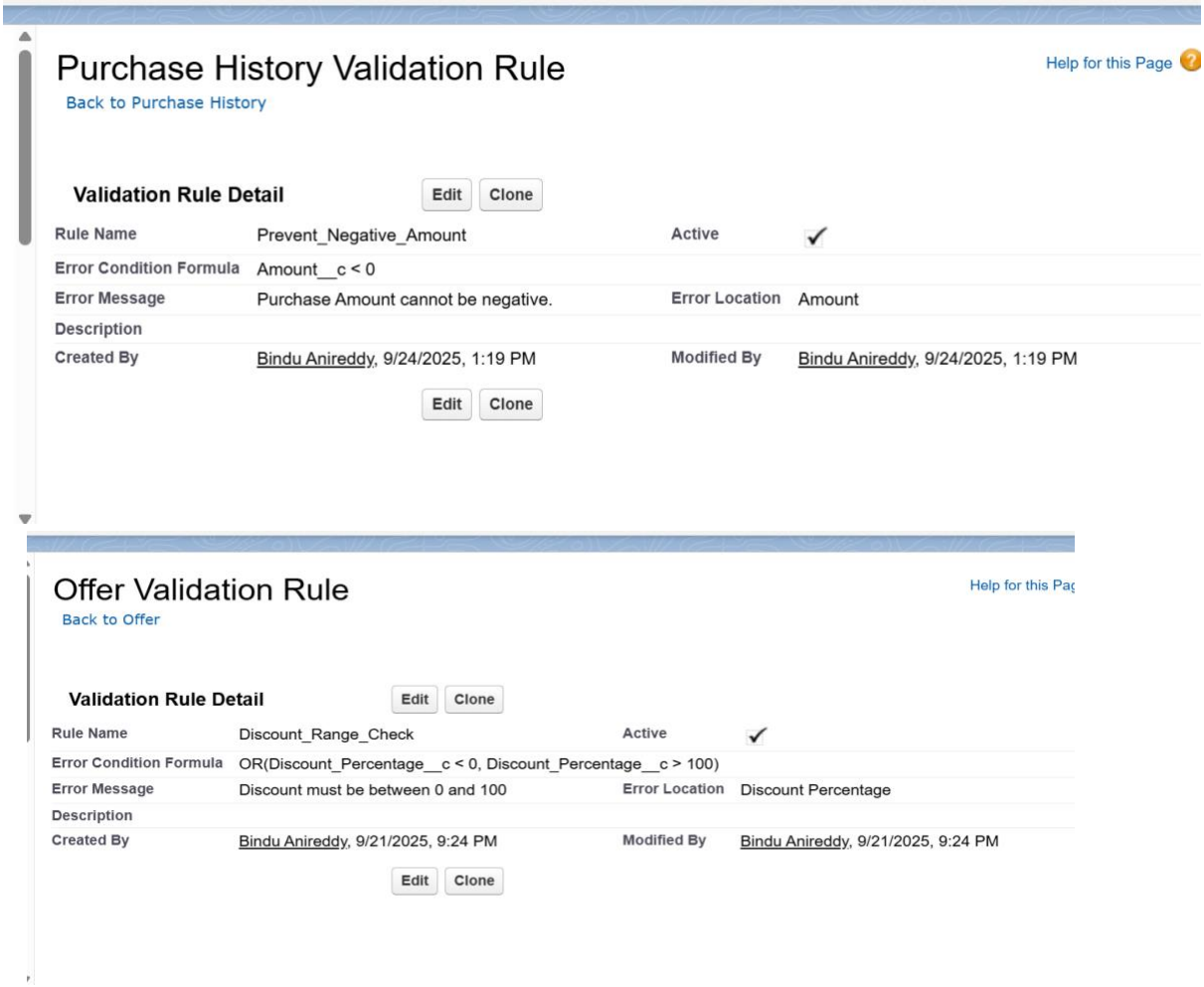


Phase 4: Process Automation (Admin)

1.Validation Rules

Use Case:
In a retail business, each purchase is recorded in the **Purchase History** object to trace customer spending. If a user accidentally enters a negative amount, it could distort sales reports, sales calculations, and loyalty point allocations. For example, recording a purchase of -500 rather of 500 would incorrectly lower total sales. The validation rule assures only valid, non-negative amounts are saved, managing accurate financial records.

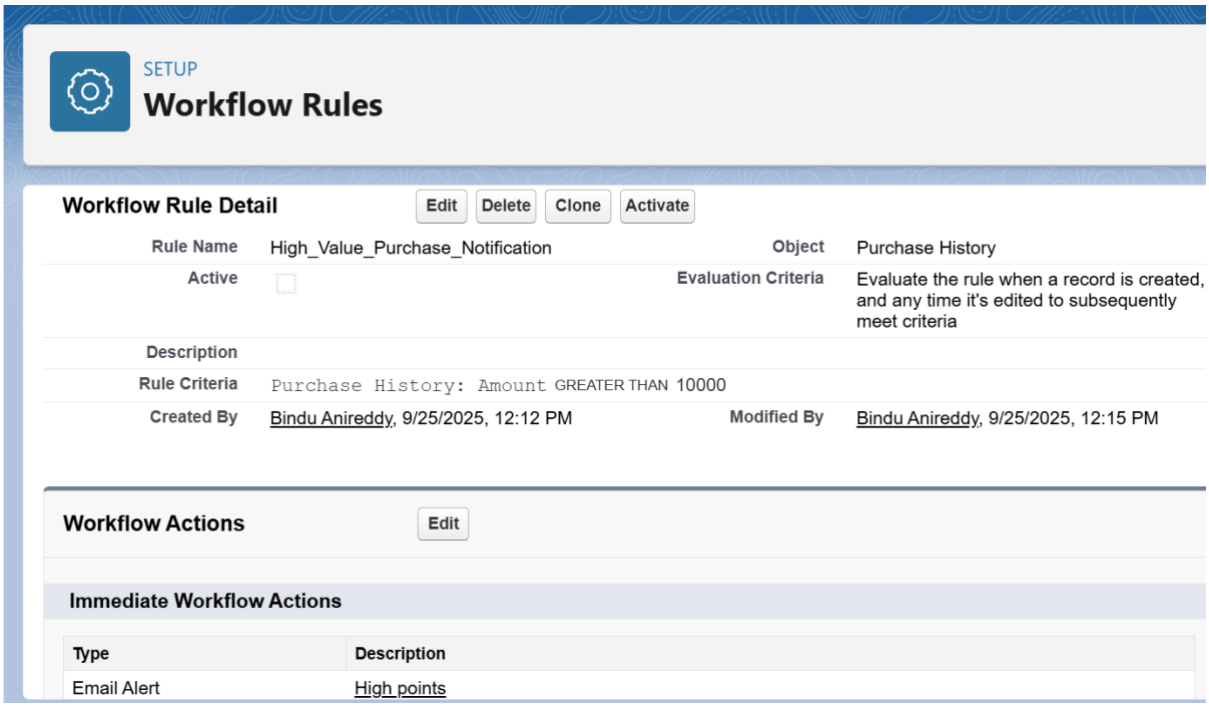
Use Case:
In retail, discounts should consistently be practical, so the **Discount Percentage** must fall between 0 and 100. This avoids errors like entering -10 or 150, which would cause incorrect billing or reporting.



2. Workflow Rules

Use Case:
If a customer makes a huge purchase above 10,000, the system instantly sends an

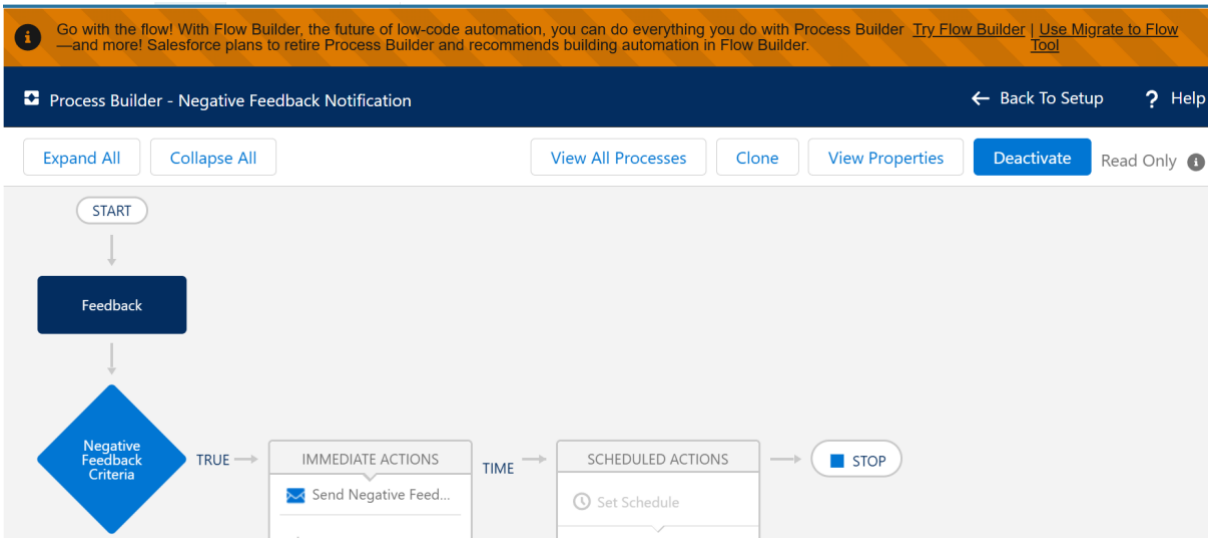
email notification to the Sales Manager. This assists the team track and review on big sales quickly without manual intervention.



3. Process Builder

Use Case:

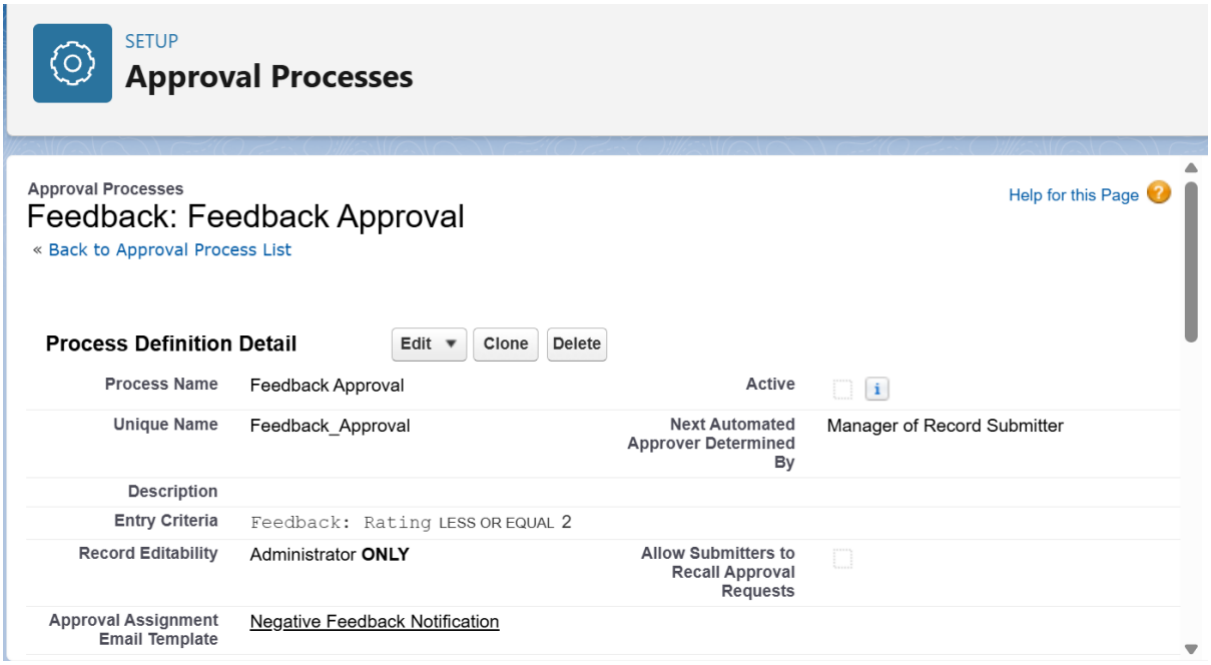
In a retail setup, if a customer provides **negative feedback**, Process Builder can automatically forward an alert to the support team. This verifies quick follow-up to solve problems and improve customer satisfaction..



4. Approval Process

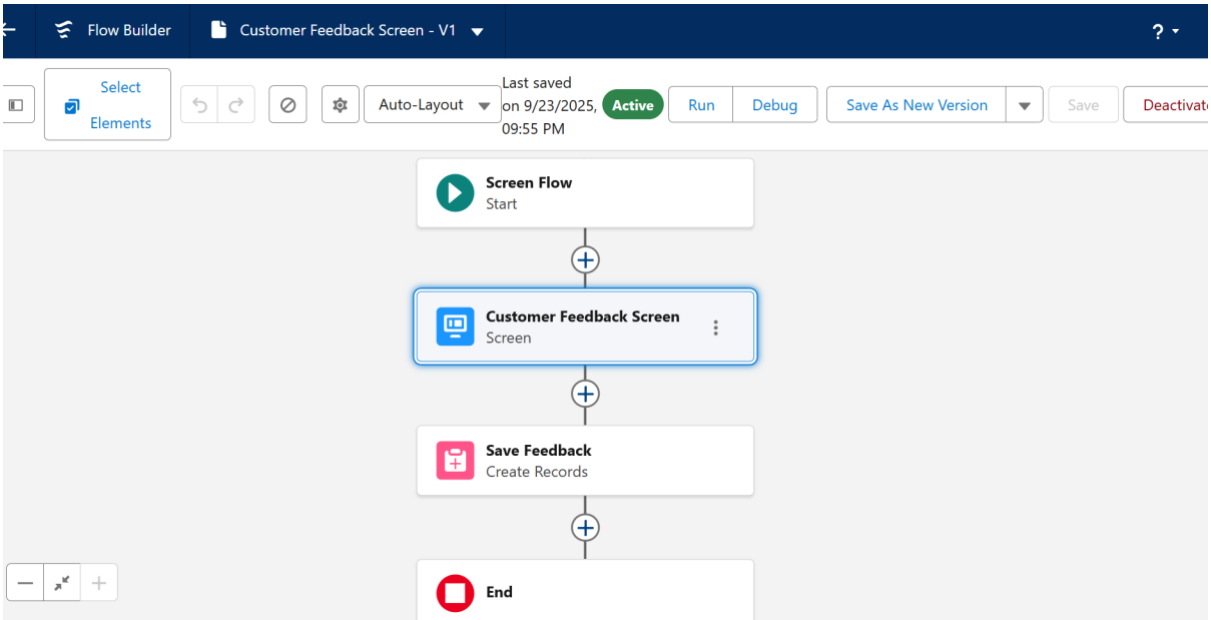
Use Case:

When a customer rating is **less than 3**, the record can be sent for manager sanction. This ensures poor experiences are examined and addressed rapidly to maintain service quality.

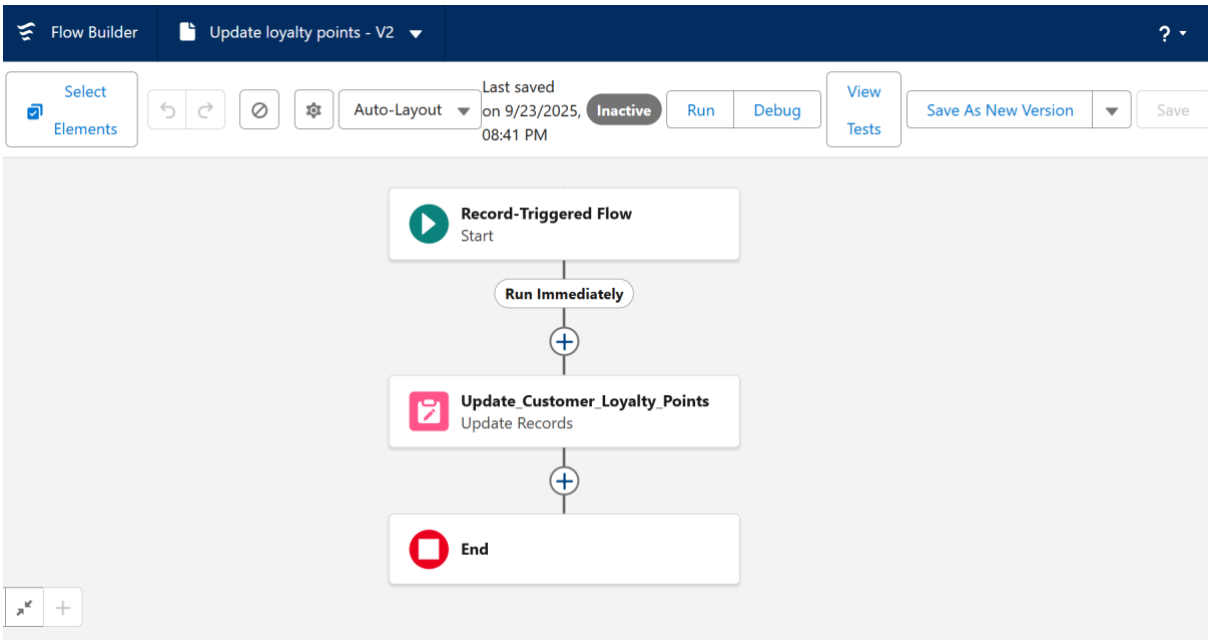


5. Flow Builder

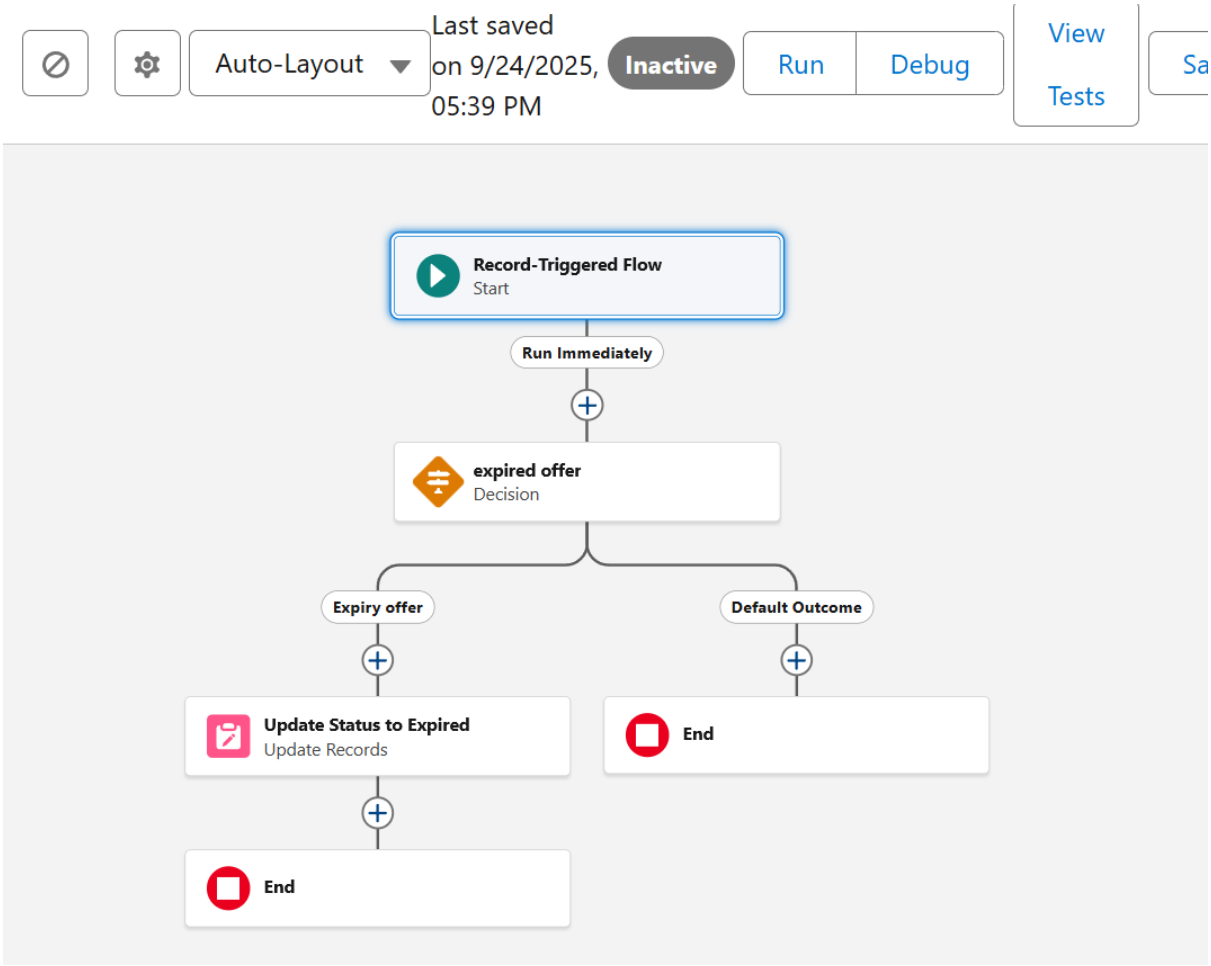
- **Screen Flow:** Design a guided flow for agents to gather customer feedback (Rating, Comments) directly while service calls.



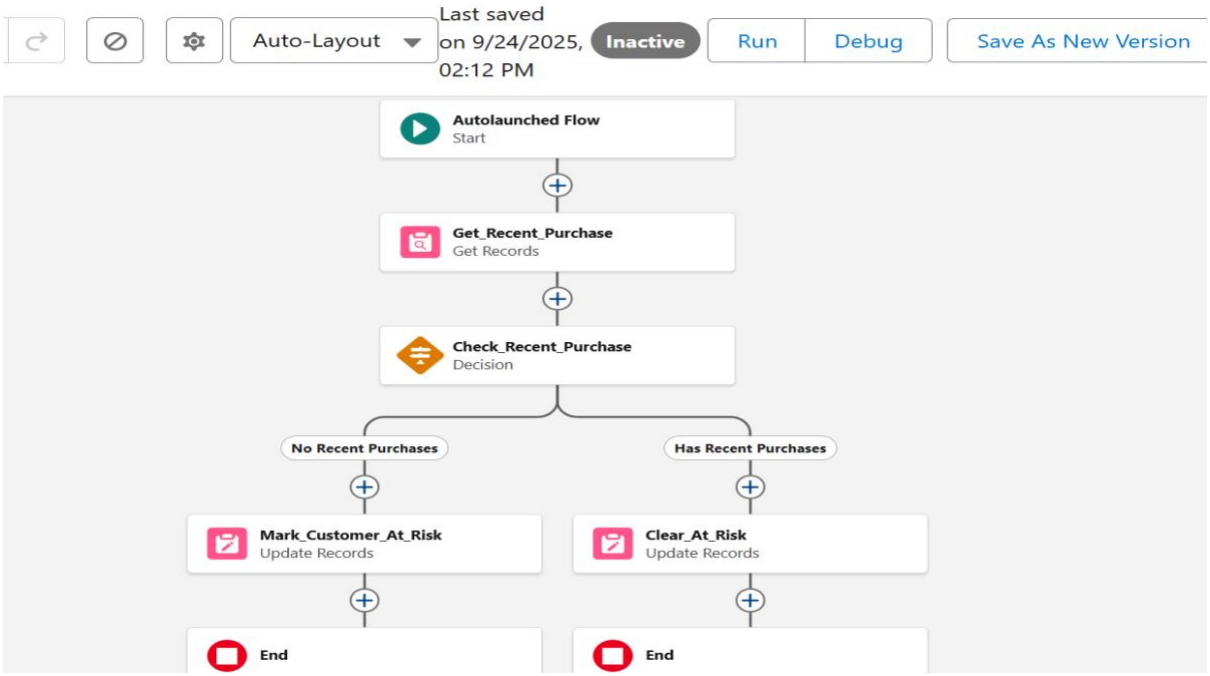
- **Record-Triggered Flow:** When a new purchase is recorded, automatically allot loyalty points to the customer.



- **Scheduled Flow:** Every Sunday night, forward a reminder email to customers whose offers are about to end.



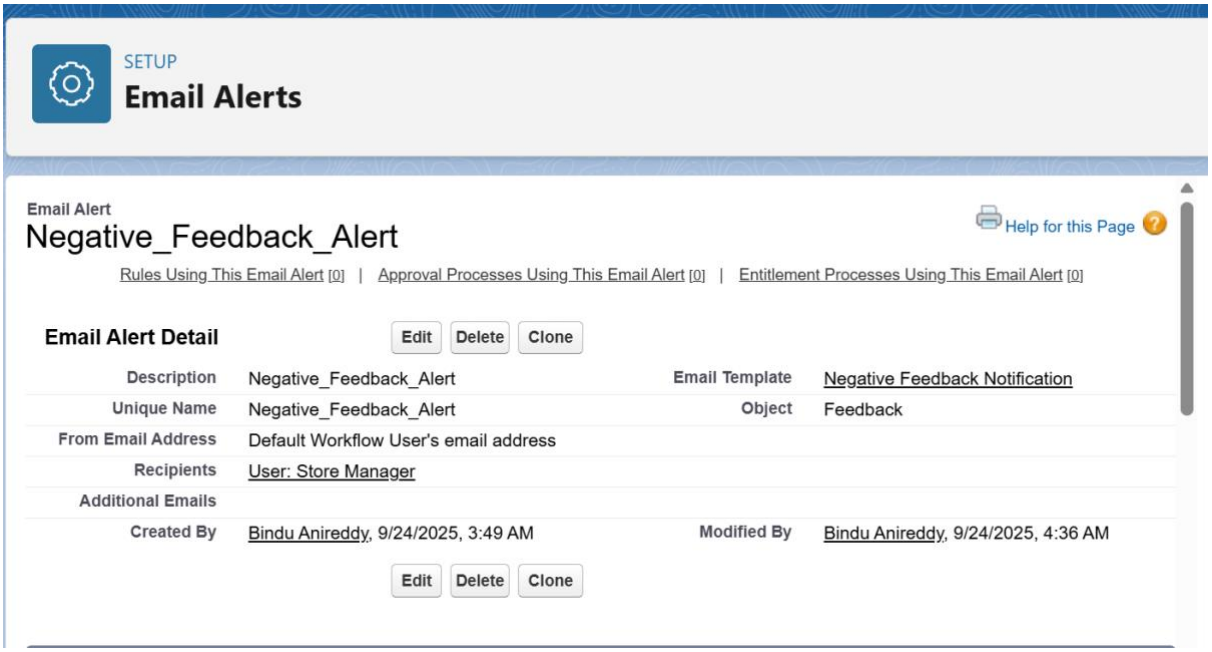
- **Auto-Launched Flow:** An Auto-launched Churn Alert Flow can trigger when a customer shows inactivity for a set period. It sends an alert to the sales team, enabling proactive engagement to reduce churn risk.

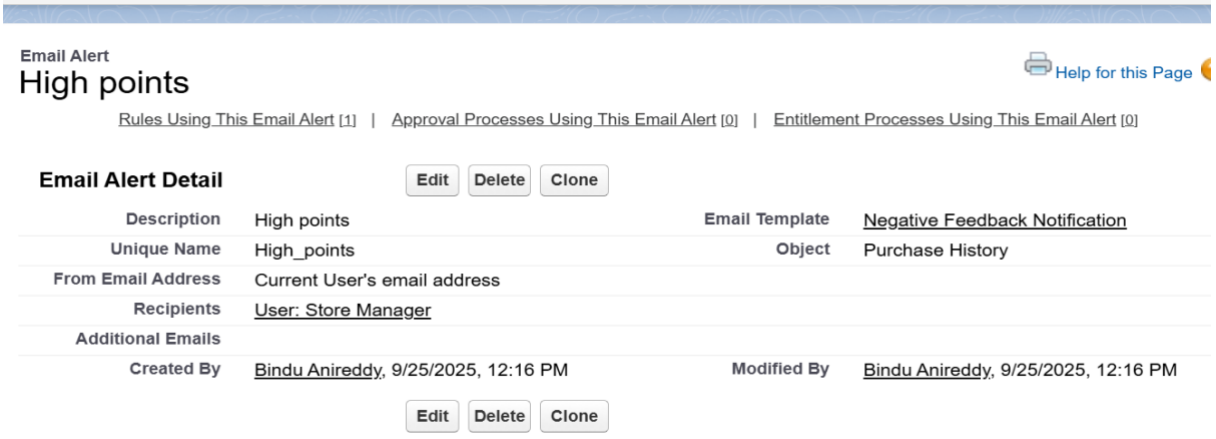


6. Email Alerts

Use Case:

When a customer enters negative feedback, an email alert can be forward to the support manager. This ensures instant action is taken to address the problem and retain customer trust.

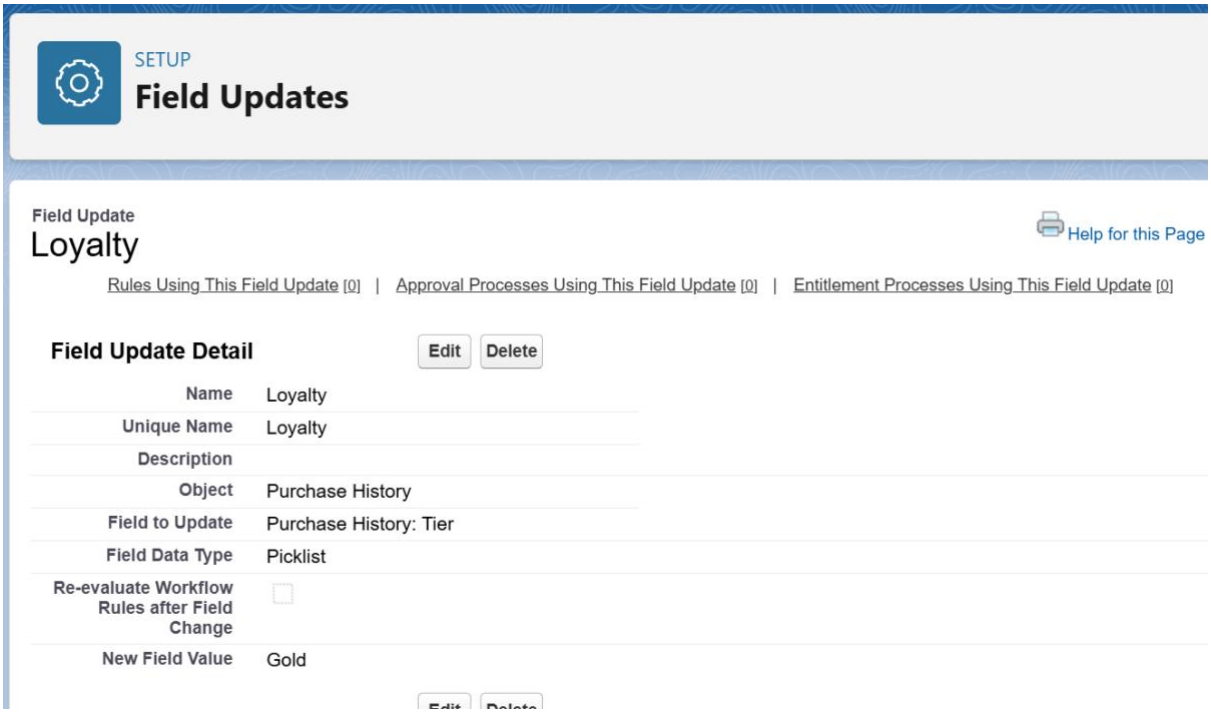




7. Field Updates

Use Case:

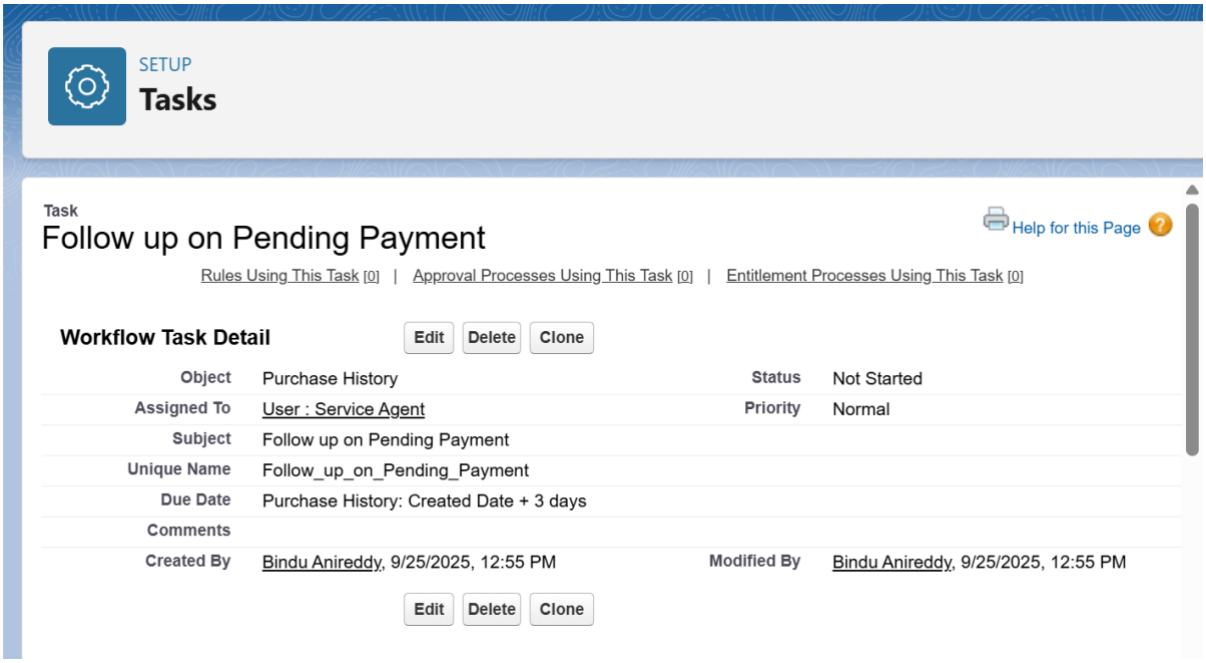
If a customer's loyalty points exceed 5000, automatically refresh their **Tier__c** field from *Silver* to *Gold*.



8. Tasks

Use Case:

Whenever a purchase has a pending payment, a task is instantly created for the sales team to assists, ensuring instant collection and minimizing overdue payments

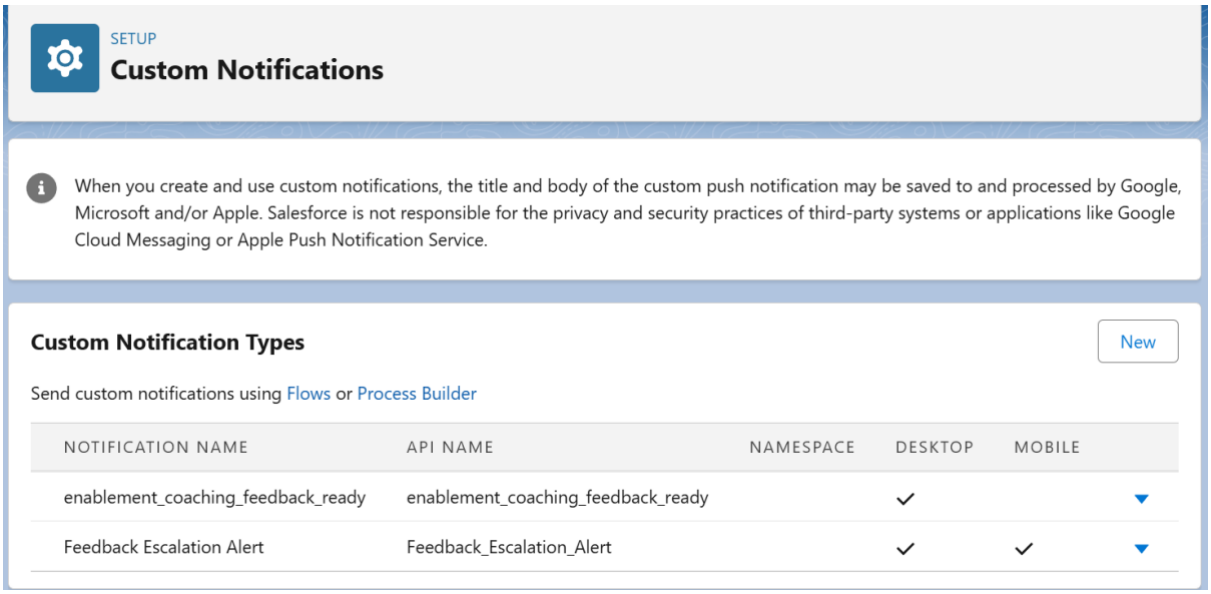


9. Custom Notifications

Use Case:

When a customer provides negative feedback, a custom notification can alert the

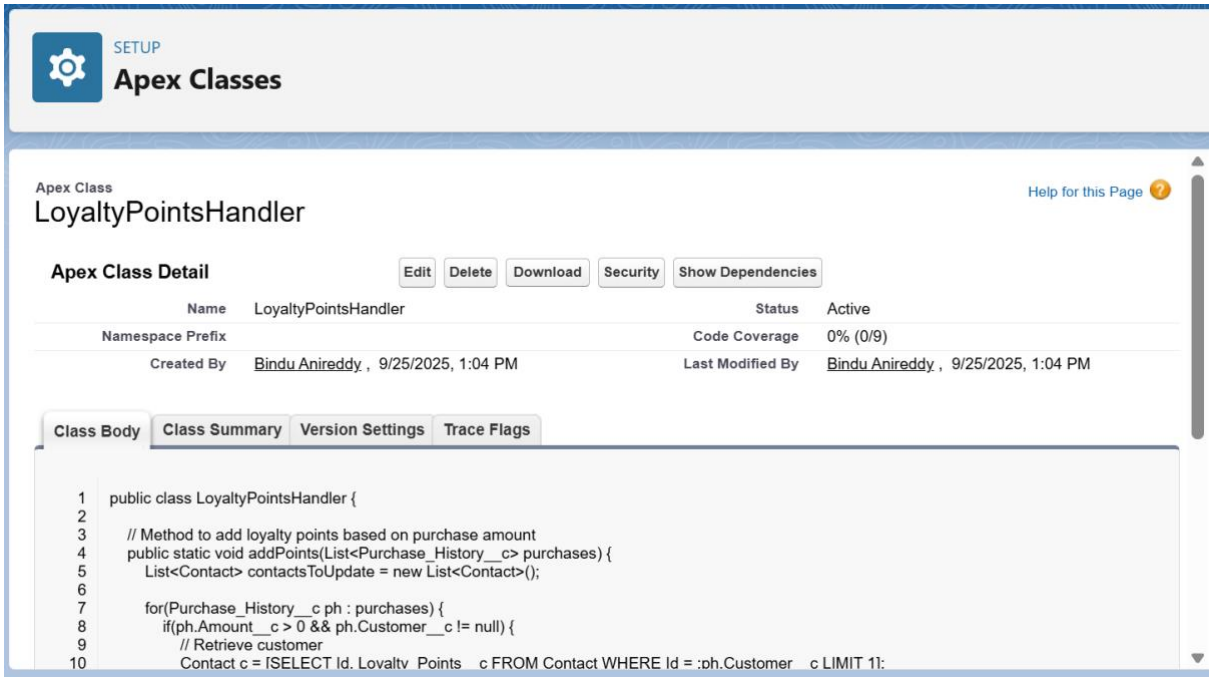
support manager automatically. This allows timely escalation and quick resolution to upgrade customer satisfaction.



Phase 5: Apex Programming (Developer)

Classes & Objects

Use Case:
When a customer creates a purchase, this class instantly calculates and adds loyalty points to their account. This supports repeat purchases by rewarding customers for their expenditure.

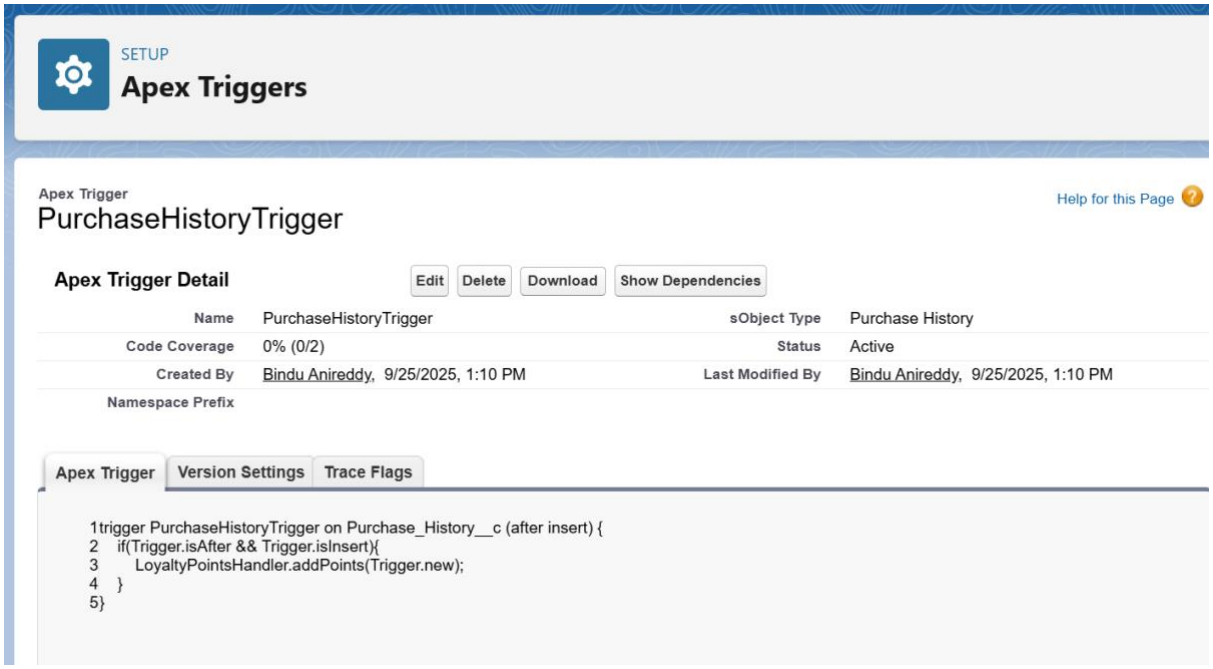


```
            Contact c = [SELECT Id, Loyalty_Points__c FROM Contact WHERE Id = :ph.Customer__c LIMIT 1];

            // Add points (1 point per 100 amount)
            c.Loyalty_Points__c = (c.Loyalty_Points__c == null ? 0 : c.Loyalty_Points__c) + Math.floor(ph.Amount__c / 100);
            contactsToUpdate.add(c);
        }
    }

    if(!contactsToUpdate.isEmpty()) {
        update contactsToUpdate;
    }
  }
}
```

Edit | Delete | Download | Security | Show Dependencies

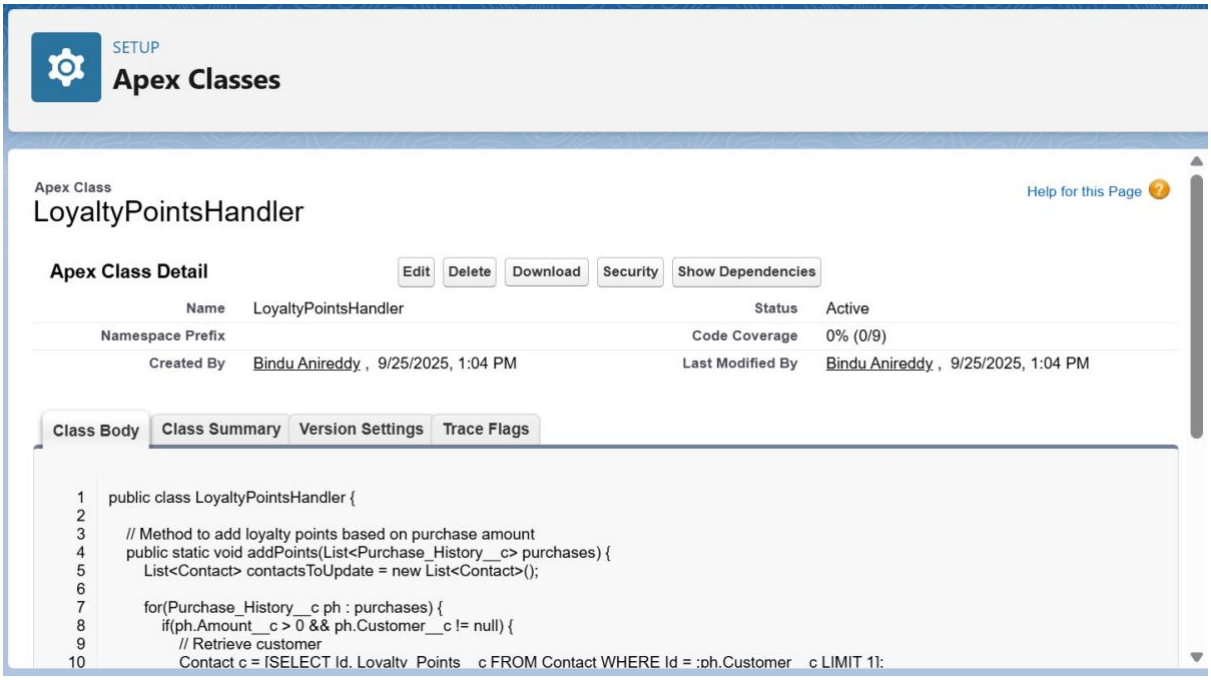
Apex Triggers (before/after insert/update/delete)

Use Case:
When a new purchase is noted, this trigger instantly updates the customer's loyalty points. It assures points are accurately added automatically, improving reward management.

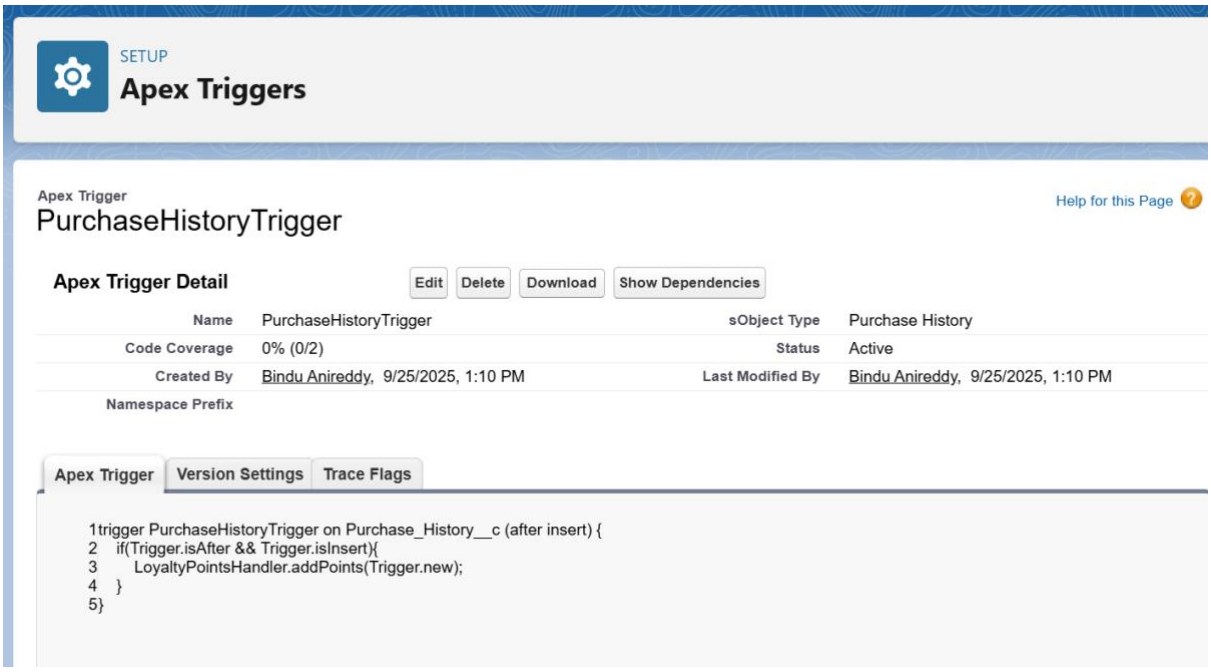


Trigger Design Pattern

Use Case:
When a new purchase is registered, the trigger deploys the handler class to upgrade the customer's loyalty points automatically. This secures clean, maintainable code while strictly processing all purchases.



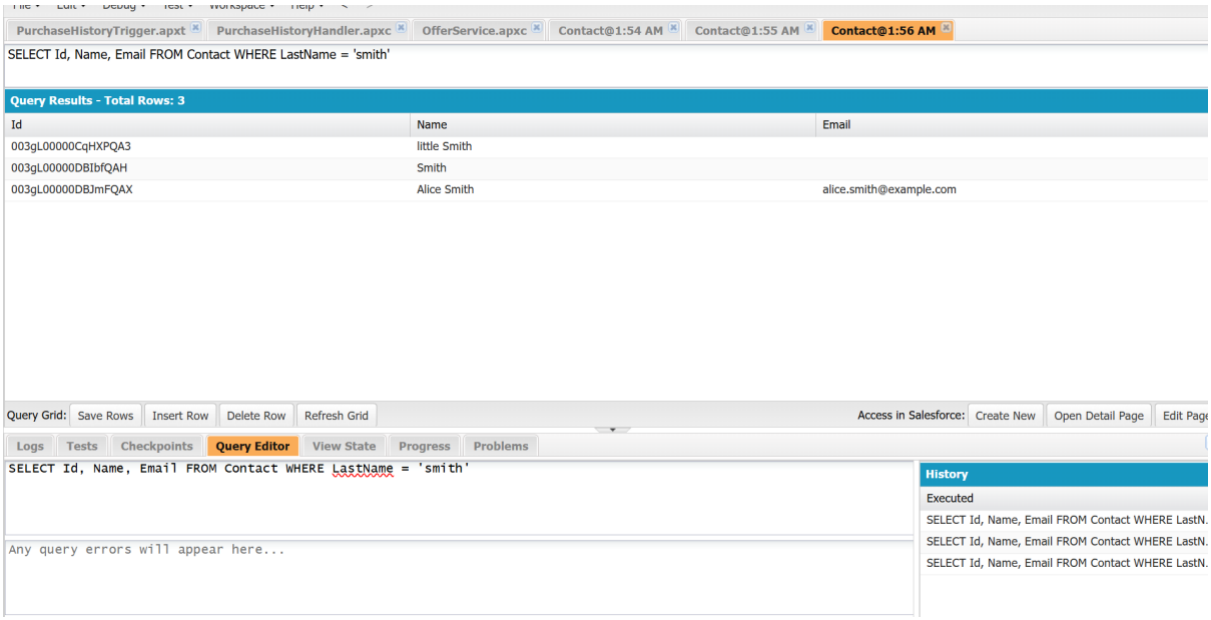

```
10            Contact c = [SELECT Id, Loyalty_Points__c FROM Contact WHERE Id = :ph.Customer__c LIMIT 1];
11
12            // Add points (1 point per 100 amount)
13            c.Loyalty_Points__c = (c.Loyalty_Points__c == null ? 0 : c.Loyalty_Points__c) + Math.floor(ph.Amount__c / 100);
14            contactsToUpdate.add(c);
15        }
16    }
17
18    if(!contactsToUpdate.isEmpty()) {
19        update contactsToUpdate;
20    }
21  }
22 }
```

SOQL & SOSL

Use Case:

SOQL can be used to fetch specific customer records, such as locating all Contacts with a specific last name for targeted follow-ups. SOSL enables quickly search over multiple objects and fields, like locating any record containing “John” in contacts, accounts, or opportunities.



Collections: List, Set, Map

Use Case:

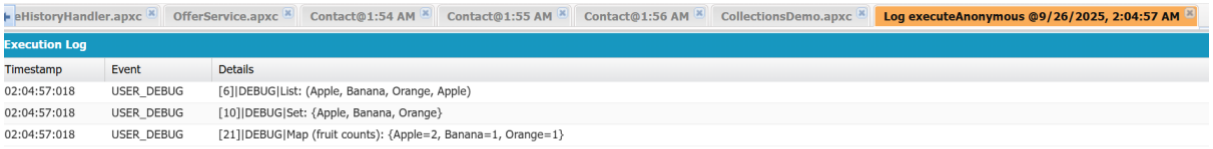
List & Set:

A customer adds products to their cart. Keeps all items in the order they were added, including duplicates (e.g., 2 Apples, 1 Banana). To show **unique items** in the cart for summary display (so you don’t show “Apple” twice in the product list).

Map:

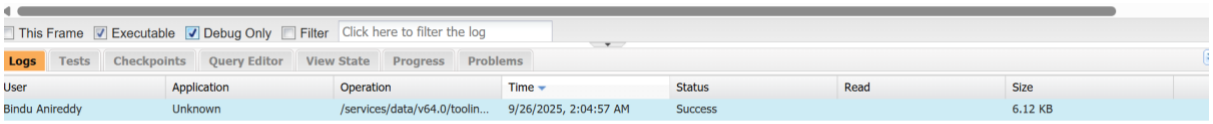
Counting how many of each product are in stock or sold. Key = Product Name, Value = Quantity. Helps quickly check stock levels or generate a report of sold items.

```apex
public class CollectionsDemo {

    public static void demoCollections() {
        // ----- LIST -----
        List<String> fruits = new List<String>{'Apple', 'Banana', 'Orange', 'Apple'};
        System.debug('List: ' + fruits);

        // ----- SET -----
        Set<String> uniqueFruits = new Set<String>{'Apple', 'Banana', 'Orange', 'Apple'};
        System.debug('Set: ' + uniqueFruits);

        // ----- MAP -----
        Map<String, Integer> fruitCount = new Map<String, Integer>();
        for(String fruit : fruits){
            if(fruitCount.containsKey(fruit)){
                fruitCount.put(fruit, fruitCount.get(fruit) + 1);
            } else {
                fruitCount.put(fruit, 1);
            }
        }
        System.debug('Map (fruit counts): ' + fruitCount);
    }
}
```



Execution Log

Timestamp	Event	Details
02:04:57:018	USER_DEBUG	[6]|DEBUG|List: (Apple, Banana, Orange, Apple)
02:04:57:018	USER_DEBUG	[10]|DEBUG|Set: {Apple, Banana, Orange}
02:04:57:018	USER_DEBUG	[21]|DEBUG|Map (fruit counts): {Apple=2, Banana=1, Orange=1}



This Frame | Executable | Debug Only | Filter | Click here to filter the log

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems

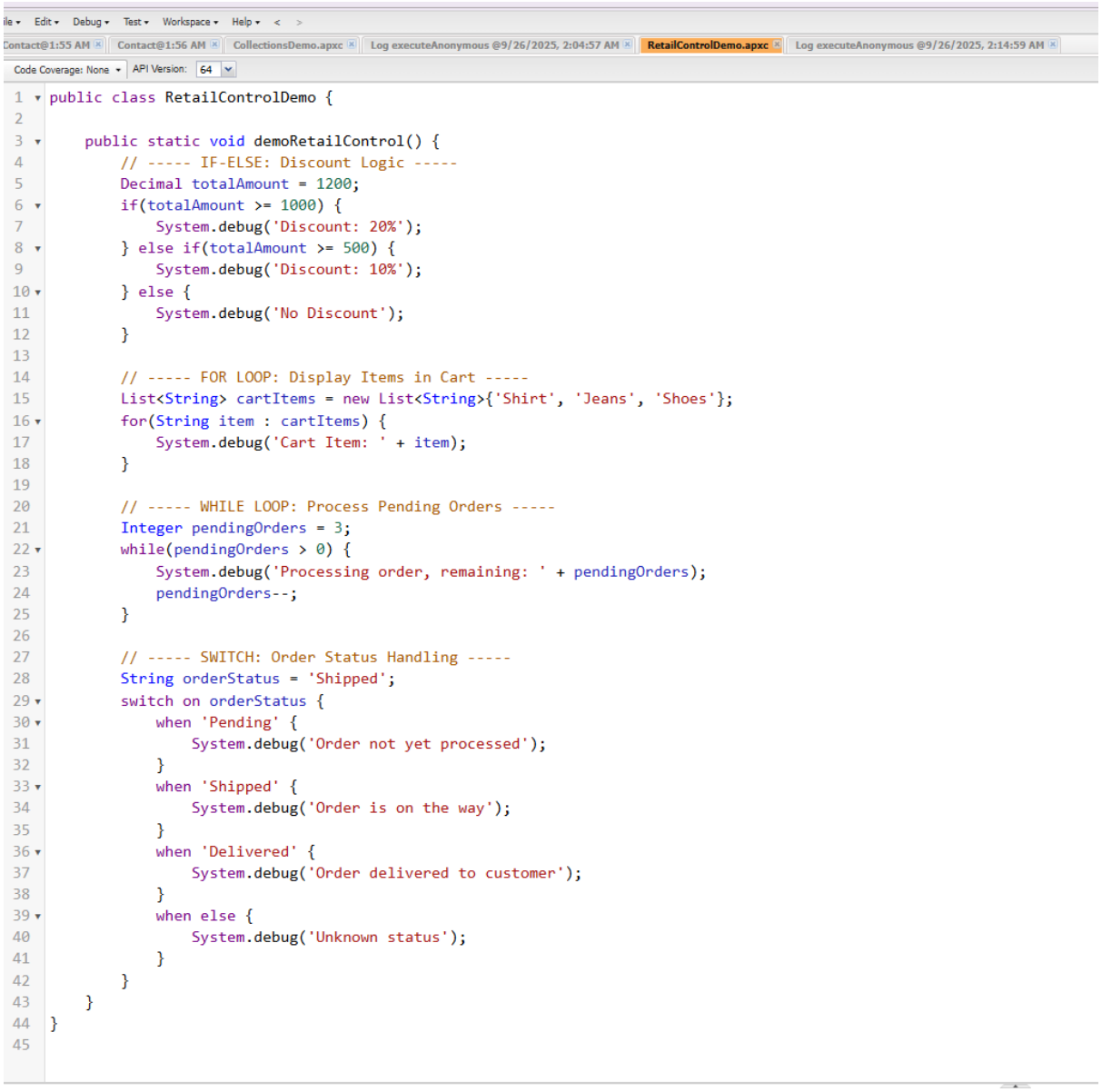
User	Application	Operation	Time	Status	Read	Size
Bindu Aniredddy	Unknown	/services/data/v64.0/toolin...	9/26/2025, 2:04:57 AM	Success		6.12 KB

Control Statements

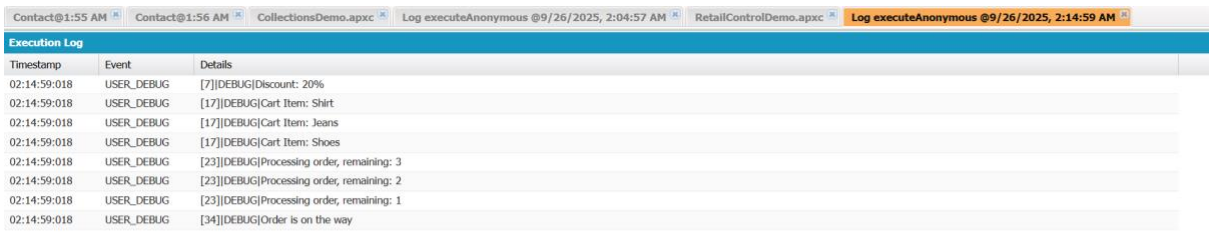
Use Case:

Automate Order Processing: Apply discounts, display cart items, and process pending orders automatically for each customer purchase.

Track Order Status: Handle different order states (Pending, Shipped, Delivered) to provide real-time updates to customers and staff.

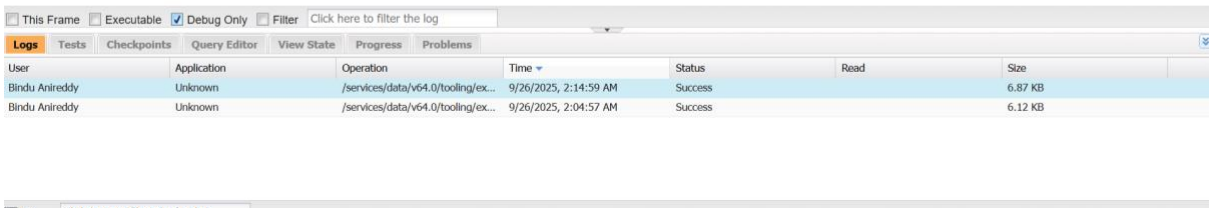


```
public class RetailControlDemo {

    public static void demoRetailControl() {
        // ----- IF-ELSE: Discount Logic -----
        Decimal totalAmount = 1200;
        if(totalAmount >= 1000) {
            System.debug('Discount: 20%');
        } else if(totalAmount >= 500) {
            System.debug('Discount: 10%');
        } else {
            System.debug('No Discount');
        }

        // ----- FOR LOOP: Display Items in Cart -----
        List<String> cartItems = new List<String>{'Shirt', 'Jeans', 'Shoes'};
        for(String item : cartItems) {
            System.debug('Cart Item: ' + item);
        }

        // ----- WHILE LOOP: Process Pending Orders -----
        Integer pendingOrders = 3;
        while(pendingOrders > 0) {
            System.debug('Processing order, remaining: ' + pendingOrders);
            pendingOrders--;
        }

        // ----- SWITCH: Order Status Handling -----
        String orderStatus = 'Shipped';
        switch on orderStatus {
            when 'Pending' {
                System.debug('Order not yet processed');
            }
            when 'Shipped' {
                System.debug('Order is on the way');
            }
            when 'Delivered' {
                System.debug('Order delivered to customer');
            }
            when else {
                System.debug('Unknown status');
            }
        }
    }
}
```



Execution Log

Timestamp	Event	Details
02:14:59:018	USER_DEBUG	[7]|DEBUG|Discount: 20%
02:14:59:018	USER_DEBUG	[17]|DEBUG|Cart Item: Shirt
02:14:59:018	USER_DEBUG	[17]|DEBUG|Cart Item: Jeans
02:14:59:018	USER_DEBUG	[17]|DEBUG|Cart Item: Shoes
02:14:59:018	USER_DEBUG	[23]|DEBUG|Processing order, remaining: 3
02:14:59:018	USER_DEBUG	[23]|DEBUG|Processing order, remaining: 2
02:14:59:018	USER_DEBUG	[23]|DEBUG|Processing order, remaining: 1
02:14:59:018	USER_DEBUG	[34]|DEBUG|Order is on the way

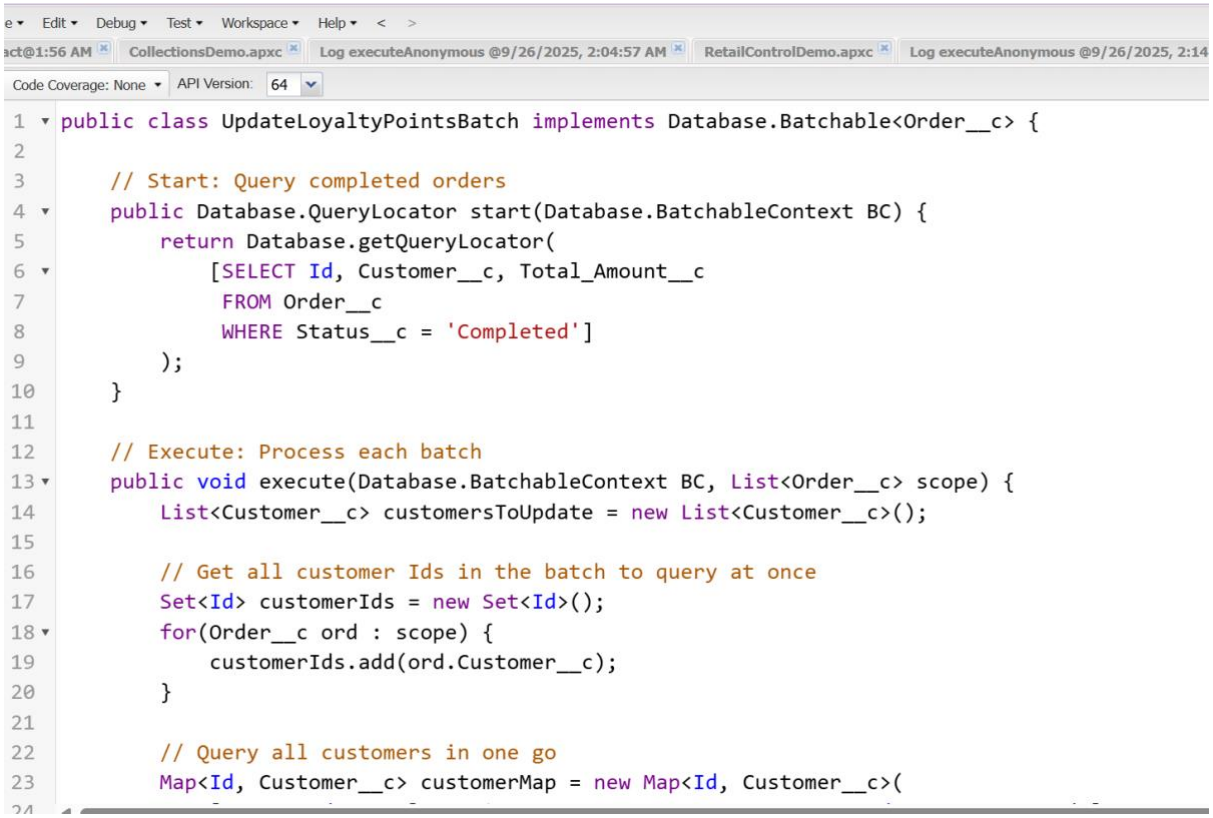


Logs

User	Application	Operation	Time	Status	Read	Size
Bindu Anireddy	Unknown	/services/data/v64.0/tooling/ex...	9/26/2025, 2:14:59 AM	Success		6.87 KB
Bindu Anireddy	Unknown	/services/data/v64.0/tooling/ex...	9/26/2025, 2:04:57 AM	Success		6.12 KB

Batch Apex

Use Case:
Automatically update customer loyalty points for all completed orders in bulk.
Process thousands of completed orders asynchronously to keep the retail system up-to-date without hitting governor limits.

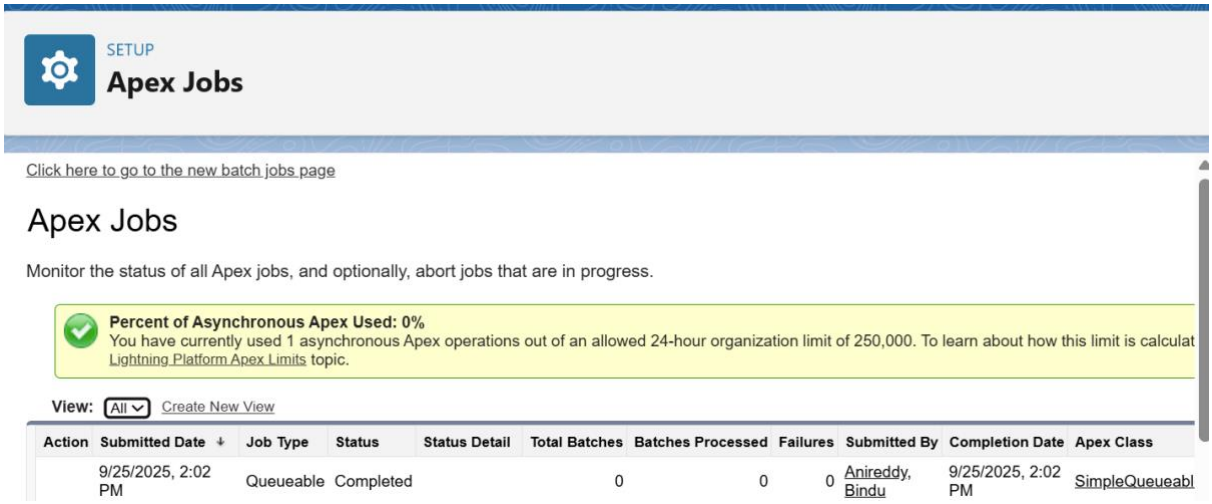



```
public class UpdateLoyaltyPointsBatch implements Database.Batchable<Order__c> {

    // Start: Query completed orders
    public Database.QueryLocator start(Database.BatchableContext BC) {
        return Database.getQueryLocator(
            [SELECT Id, Customer__c, Total_Amount__c
             FROM Order__c
             WHERE Status__c = 'Completed']
        );
    }

    // Execute: Process each batch
    public void execute(Database.BatchableContext BC, List<Order__c> scope) {
        List<Customer__c> customersToUpdate = new List<Customer__c>();

        // Get all customer Ids in the batch to query at once
        Set<Id> customerIds = new Set<Id>();
        for(Order__c ord : scope) {
            customerIds.add(ord.Customer__c);
        }

        // Query all customers in one go
        Map<Id, Customer__c> customerMap = new Map<Id, Customer__c>(
```

Queueable Apex

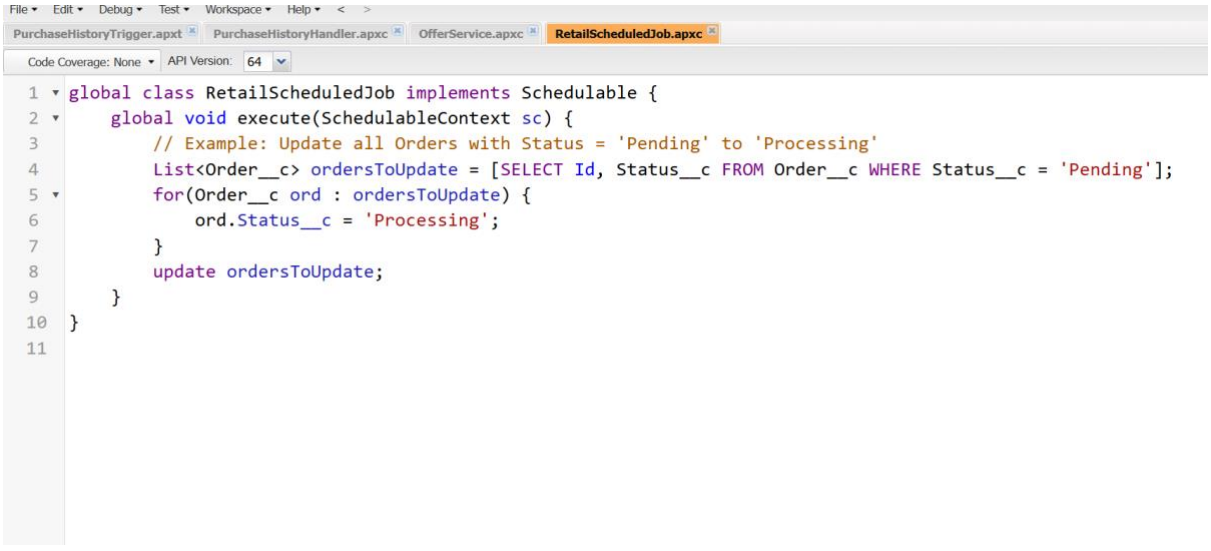
Use Case:
Asynchronous Order Updates: Update order records in the background without blocking user transactions.
Email Notifications: Send order confirmation or promotional emails to customers asynchronously after a purchase.

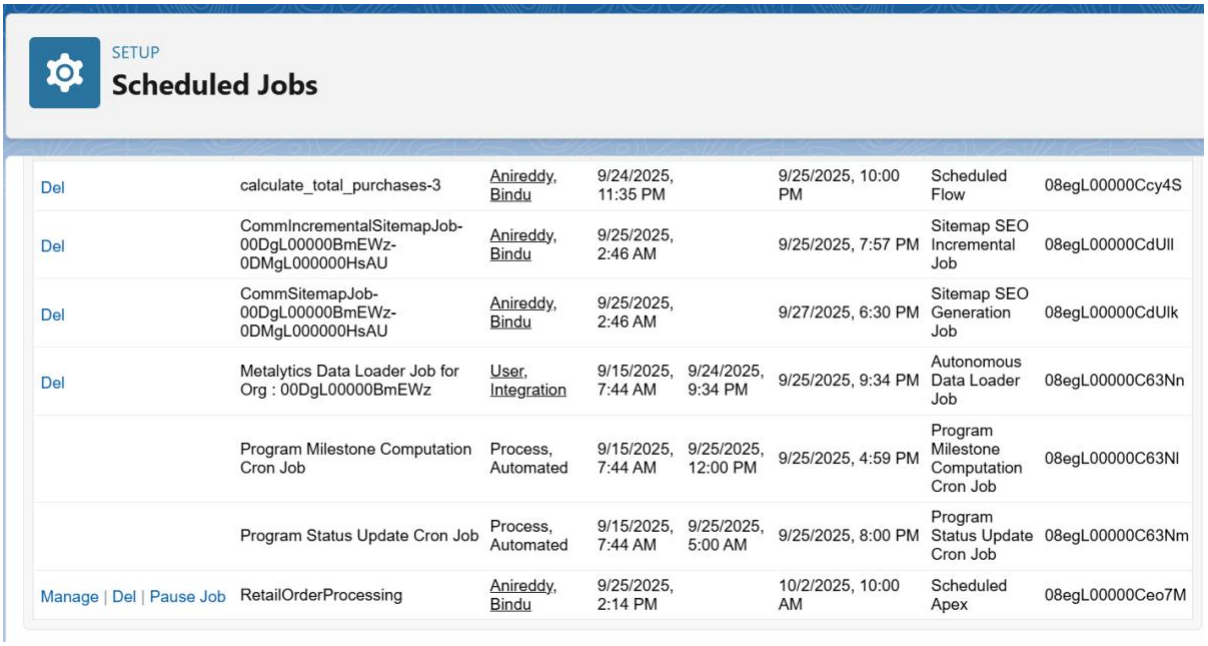
```
public class SimpleQueueable implements Queueable {

    public void execute(QueueableContext context) {
        // Simple retail example: log a message
        System.debug('Queueable Job Running: Updating retai
    }
}
```



Scheduled Apex

Use Case:
Automatically updates pending orders to processing every day without manual intervention.





Future Methods

Use Case:

Update massive sets of orders or perform time-consuming functions asynchronously without blocking the main thread.

```
public class RetailFutureJob {

    // @future annotation allows this method to run asynchronously
    @future
    public static void updateOrderStatus(List<Id> orderIds) {
        List<Order__c> ordersToUpdate = [SELECT Id, Status__c FROM Order__c WHERE Id IN :orderIds];
        for(Order__c ord : ordersToUpdate) {
            ord.Status__c = 'Processed';
        }
        update ordersToUpdate;
    }
}
```

Exception Handling

Use Case:

Avoid a bulk update or integration from failing completely if some records have problems, while logging the error for review.

```apex
public class RetailExceptionDemo {
    public static void processOrders() {
        try {
            // Fetch pending orders
            List<Order__c> orders = [SELECT Id, Status__c FROM Order__c WHERE Status__c = 'Pending'];

            // Use System.assert to check if orders exist
            System.assert(!orders.isEmpty(), 'No pending orders found.');

            // Update orders
            for(Order__c ord : orders) {
                ord.Status__c = 'Processing';
            }
            update orders;

            System.debug('Orders updated successfully.');

        } catch (Exception e) {
            System.debug('Handled Exception: ' + e.getMessage());
        } finally {
            System.debug('Process completed (finally block executed).');
        }
    }
}
```

Test Classes

Use Case:

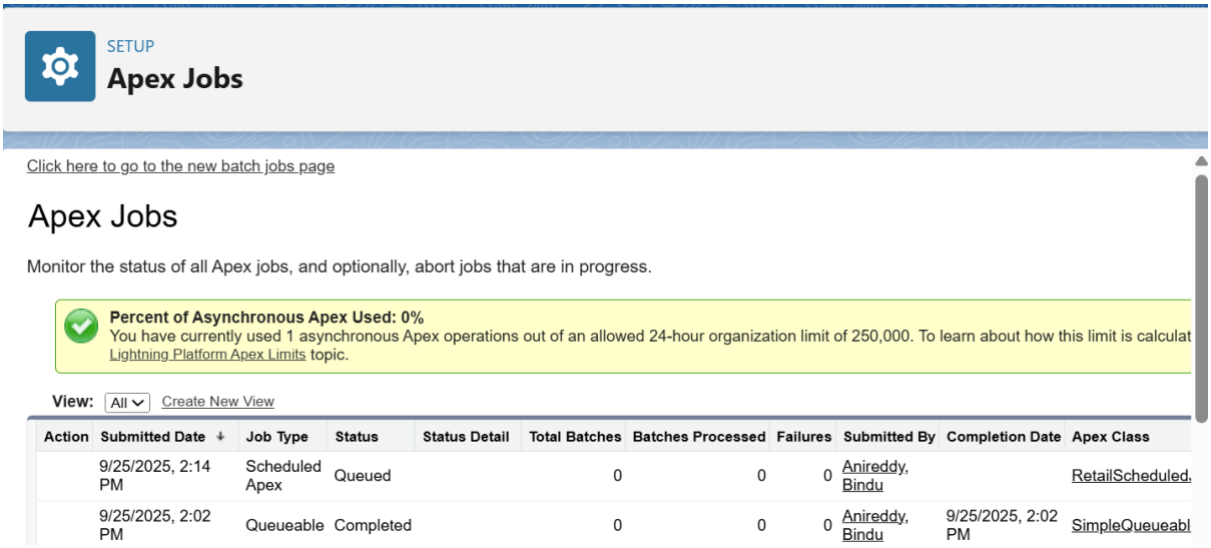
Ensures your methods like RetailExceptionDemo.processOrders() work as expected and meet Salesforce code coverage requirements.

Code Coverage: None API Version: 64

```apex
@isTest
public class RetailOrderTest {

    @isTest
    static void testProcessOrders() {
        // Create test data
        Order__c testOrder = new Order__c(Name='Test Order', Status__c='Pending');
        insert testOrder;

        // Call the method to test
        RetailExceptionDemo.processOrders();

        // Verify results
        testOrder = [SELECT Status__c FROM Order__c WHERE Id = :testOrder.Id];
        System.assertEquals('Processing', testOrder.Status__c, 'Order status should be updated to Processing');
    }
}
```

Asynchronous Processing

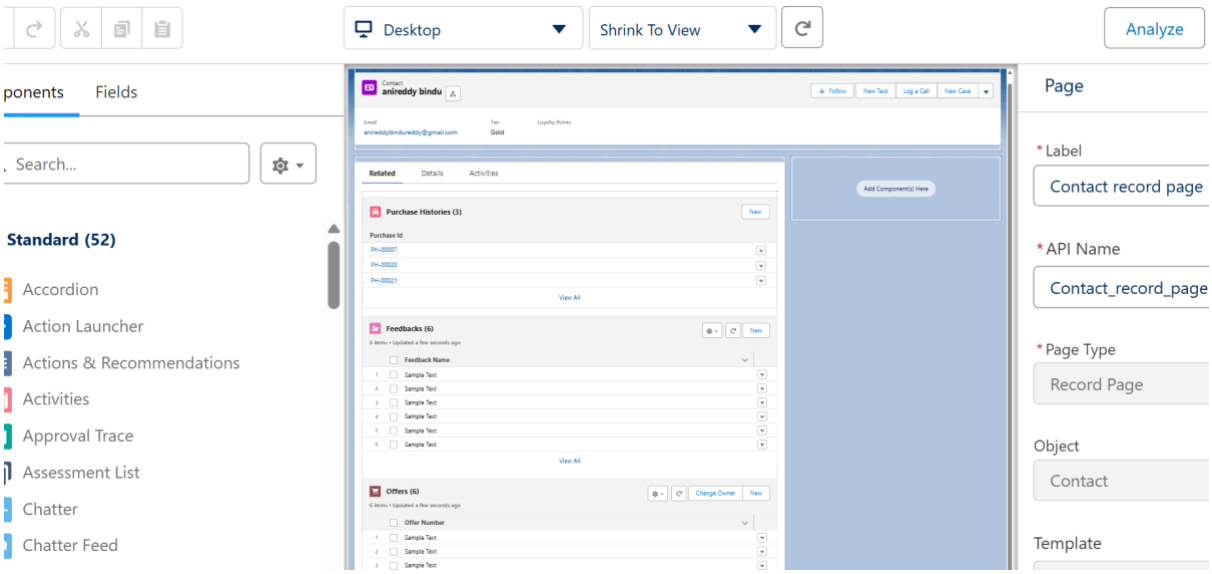
Use Case:
Automatically update large volumes of pending orders or send customer notifications in the background, ensuring users aren't blocked and Salesforce governor limits are not exceeded.



Phase 6: User Interface Development

Lightning App Builder

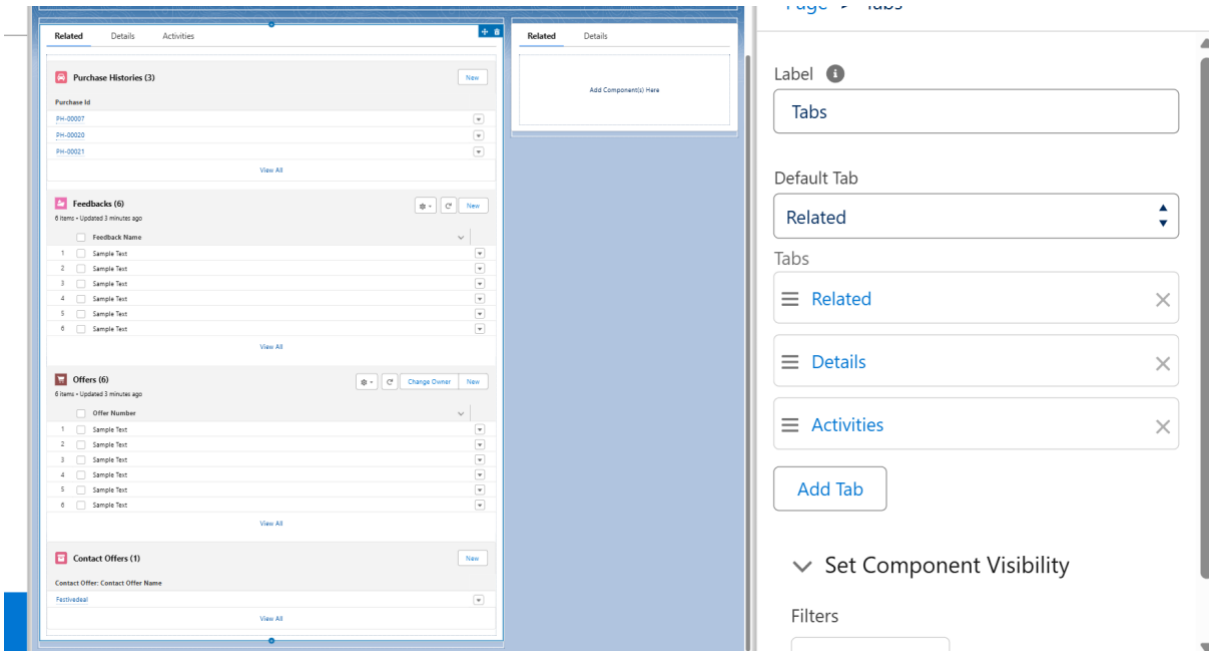
Use Case:
Provide a **360° view of a customer** by displaying their contact info, purchase history, and recent orders on a single page, enabling sales reps to act quickly without navigating multiple pages.



Tabs

Use Case:

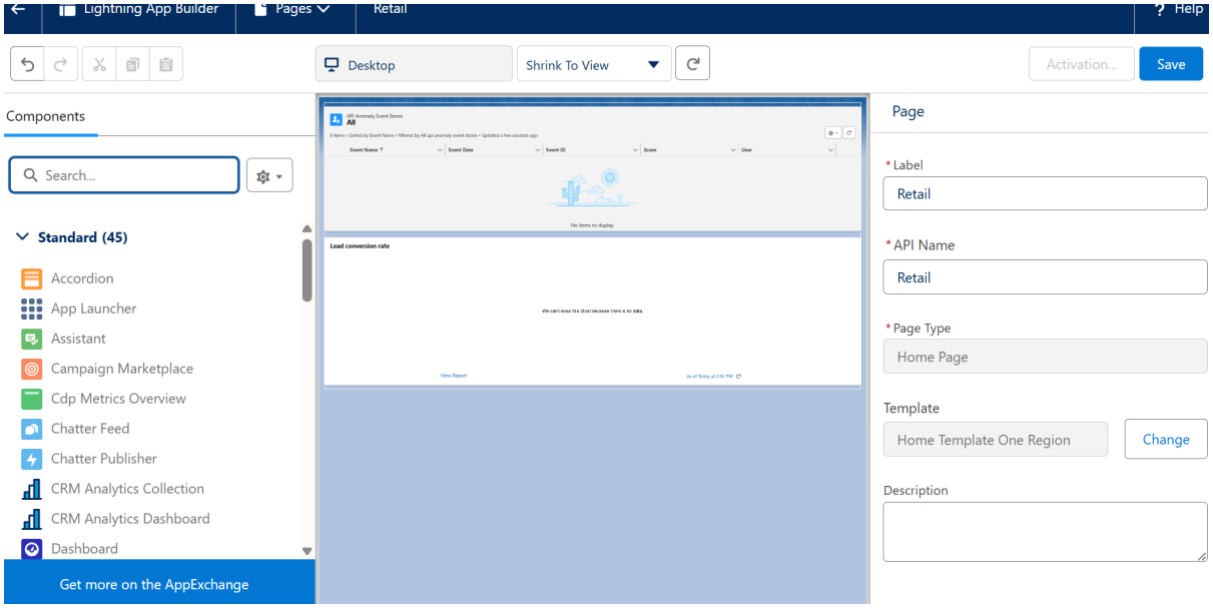
Organize related information like Orders, Support Cases, and Loyalty Points into different tabs on a customer record, enables sales reps to fast access all related data without scrolling through a single long page



Home Page

Use Case:

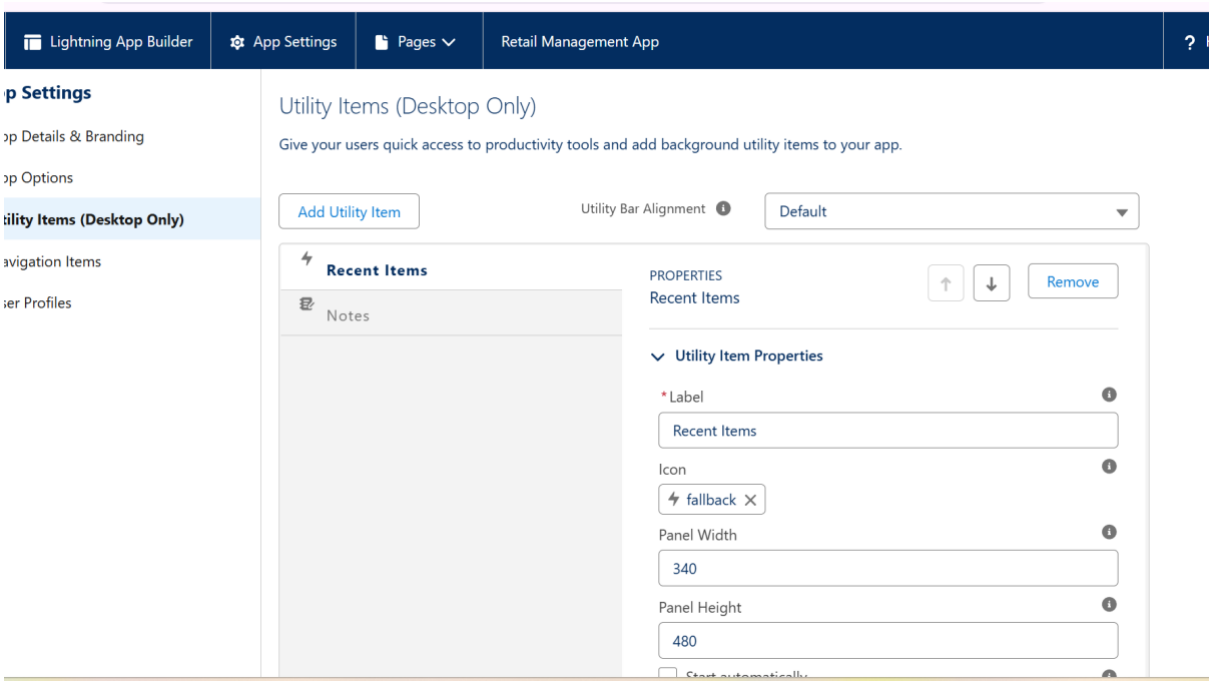
Provide sales reps and managers with a quick overview of key retail metrics, like pending orders, high-value customers, and sales performance, right when they log in.



Utility Bar

Use Case:

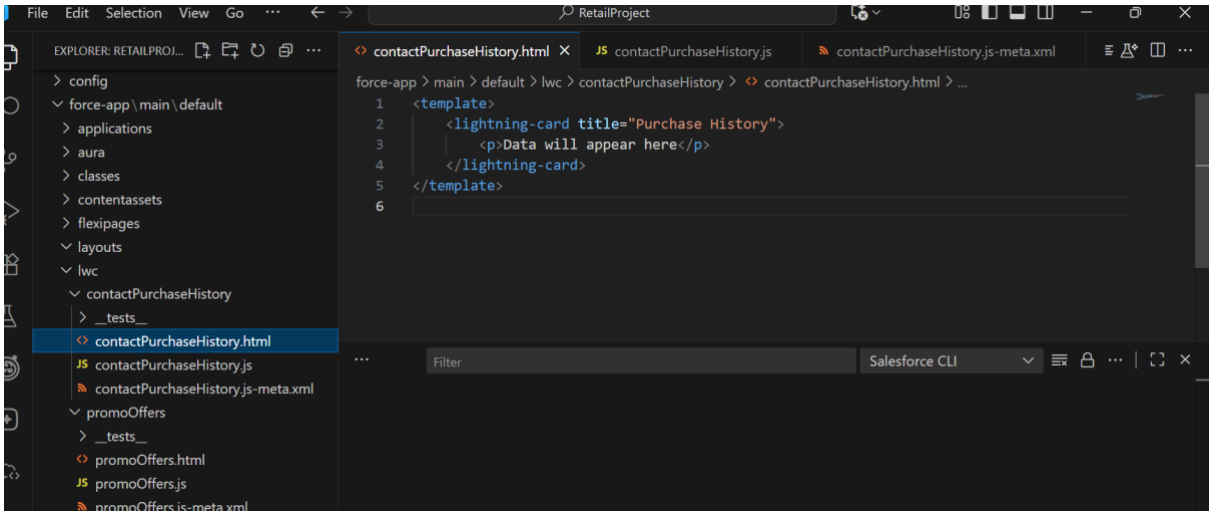
Provide sales reps with **quick access to notes, recent orders, or custom retail tools** without navigating away from the current record, improving efficiency during customer interactions.

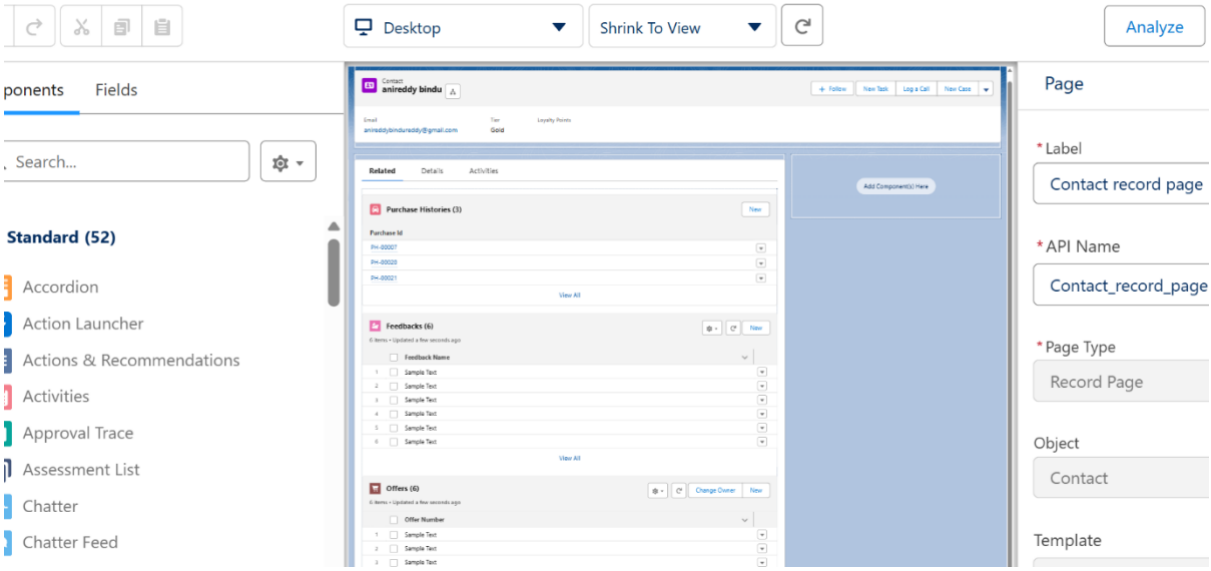


LWC (Lightning Web Components)

Use Case:

Display dynamic purchase history, order details, or loyalty points for customers in a visually interactive component, enabling sales reps to view critical information quickly.

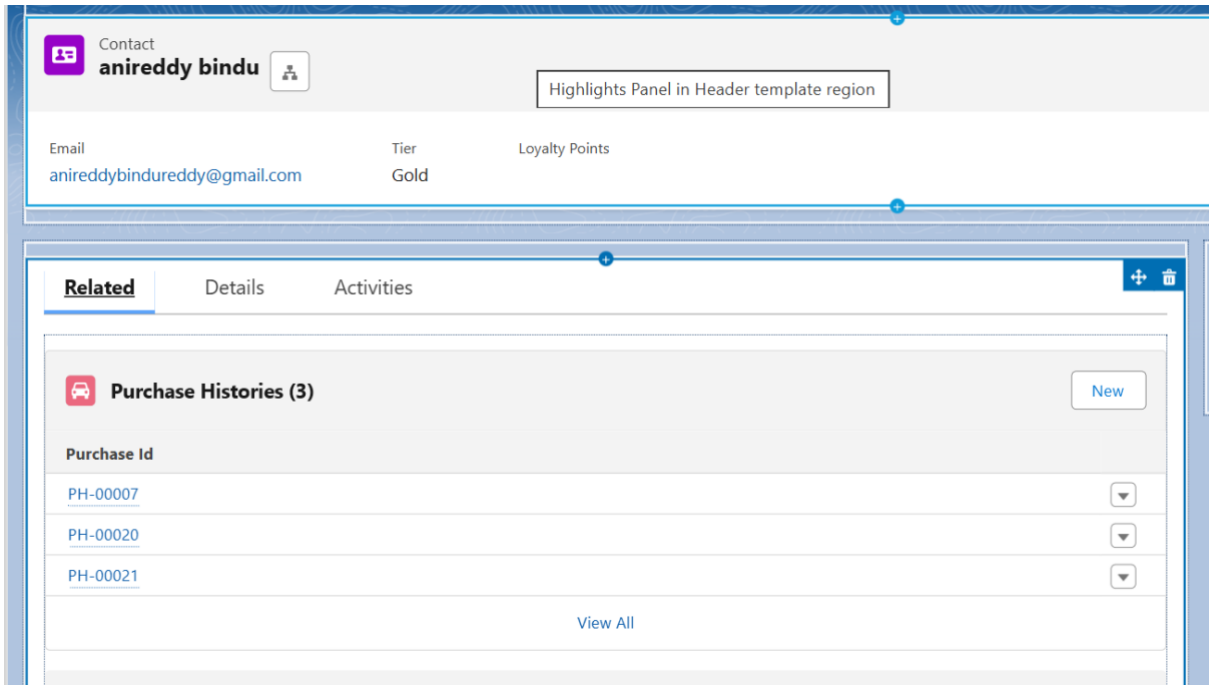




Apex with LWC

Use Case:

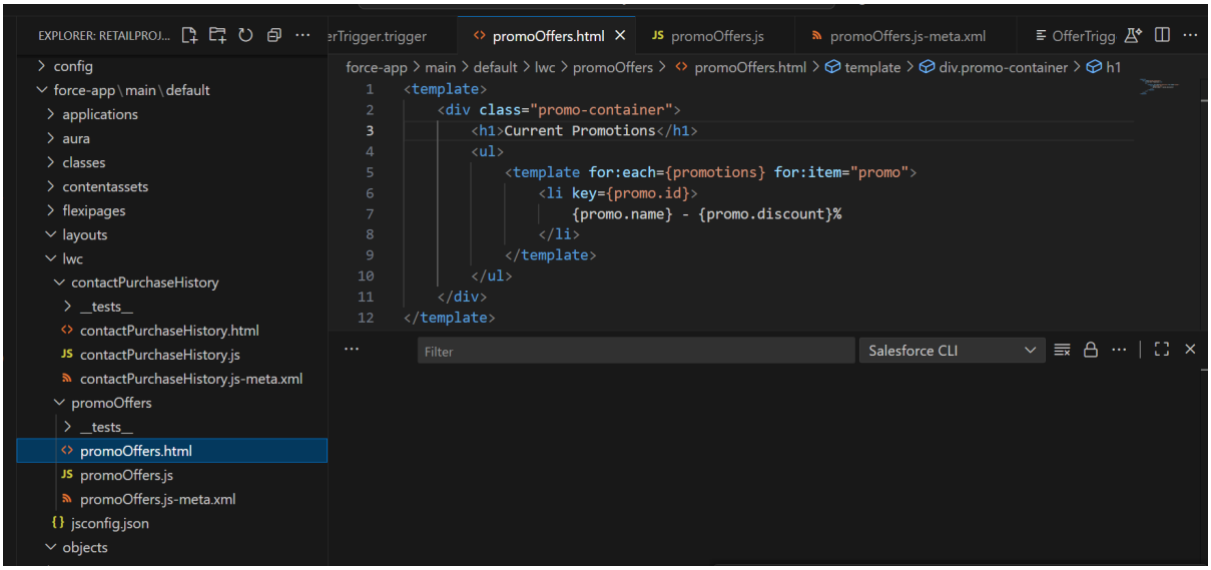
Show a customer's **real-time purchase history** dynamically on their record, combining Apex data retrieval with interactive Lightning Web Component display.

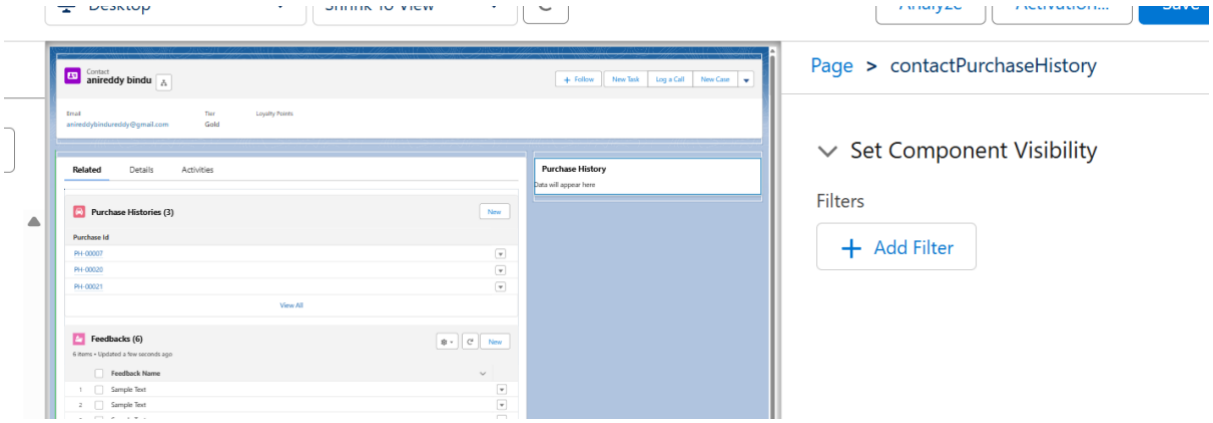


Events in LWC

Use Case:

Enable **dynamic communication between components**, such as notifying a dashboard or order summary component when a new purchase is recorded in a child component, without page reload.





Imperative Apex calls

Use Case:

Use imperative Apex calls in LWC when you need **to fetch or update data on demand**, such as retrieving a customer's latest orders after a button click.

Enable sales reps to **trigger specific actions**, like applying discounts or marking orders as shipped, without waiting for page reloads.

Provides **flexible, real-time interaction** between the UI and server-side logic for dynamic retail operations.

Navigation Service

Use Case:

Use Navigation Service in LWC to **programmatically navigate** sales reps to different records, related lists, or external URLs.

For example, after creating an order, automatically redirect the user to the **Order Detail page** or a **related Contact record**.

This improves workflow efficiency by **reducing clicks and keeping users focused** on retail operations.

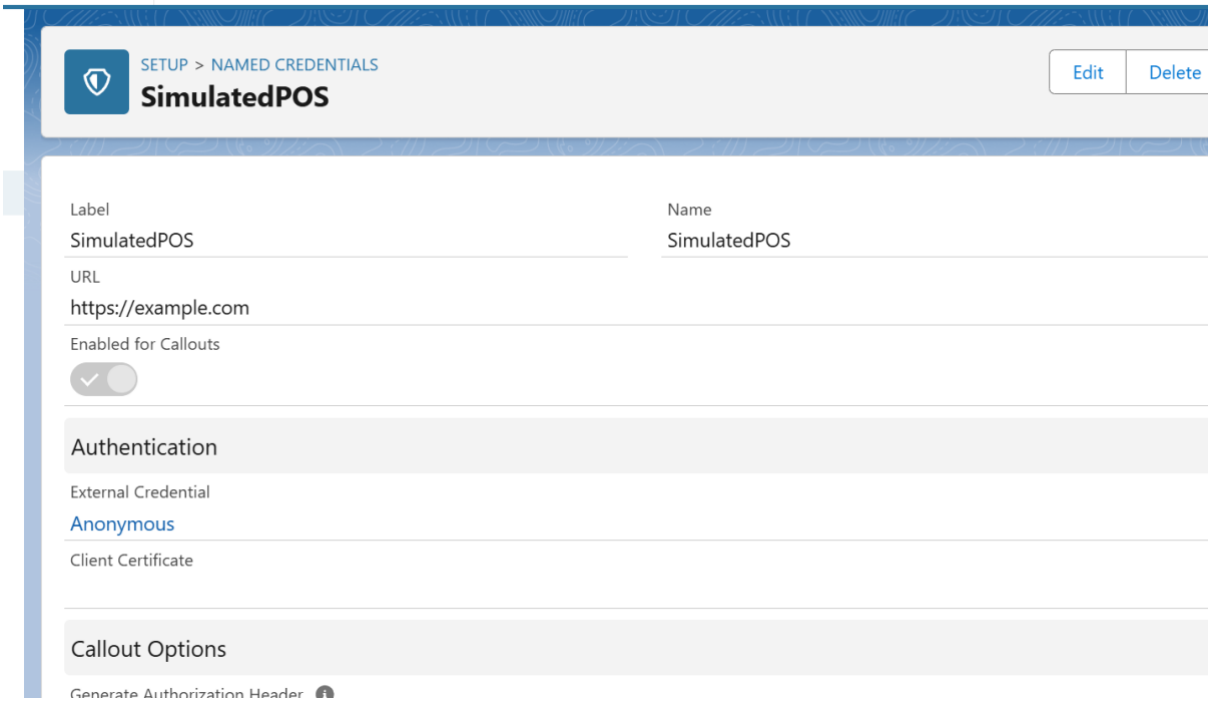
Phase 7: Integration & External Access

Named Credentials

Use Case:

Securely connect Salesforce to external systems like payment gateways or inventory APIs, enabling real-time data access.

Eliminates storing credentials in code while simplifying authentication for retail integrations.



External Services

Use Case:

Integrate Salesforce with external APIs, like shipping providers or payment gateways, to automatically process orders and track shipments.

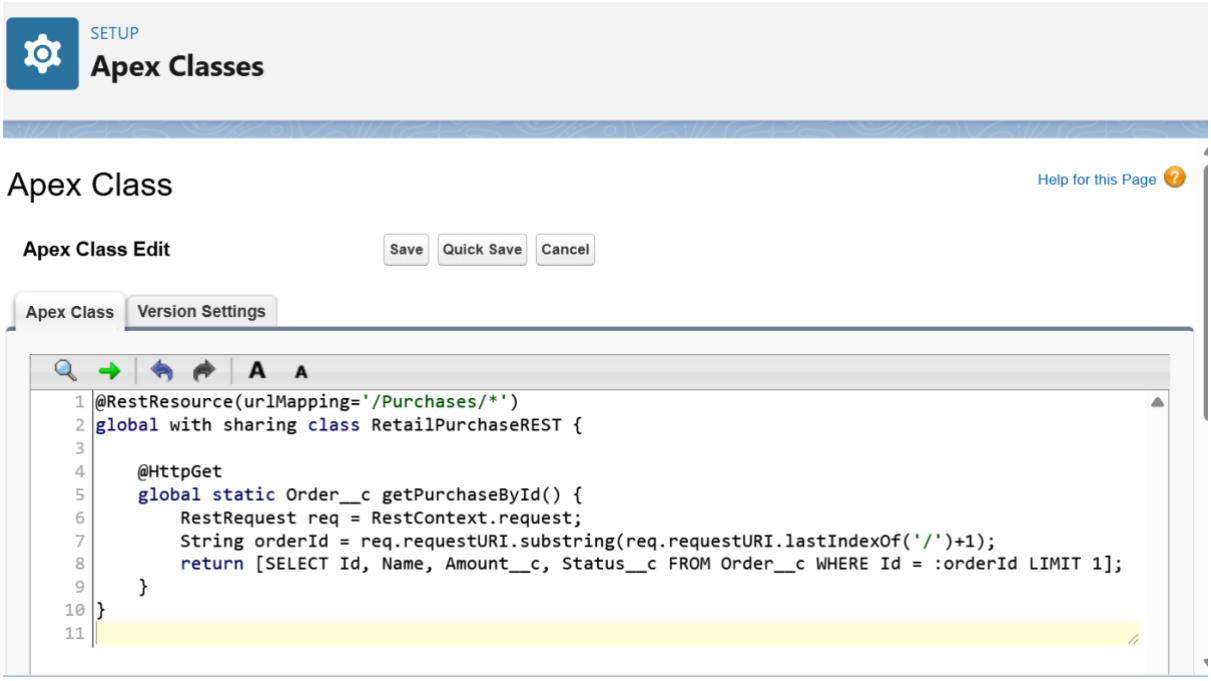
Enables sales reps to access external data and services directly within Salesforce without custom Apex code.

Web Services (REST/SOAP)

Use Case:

REST: Fetch order details or update status for external apps like mobile retail dashboards.

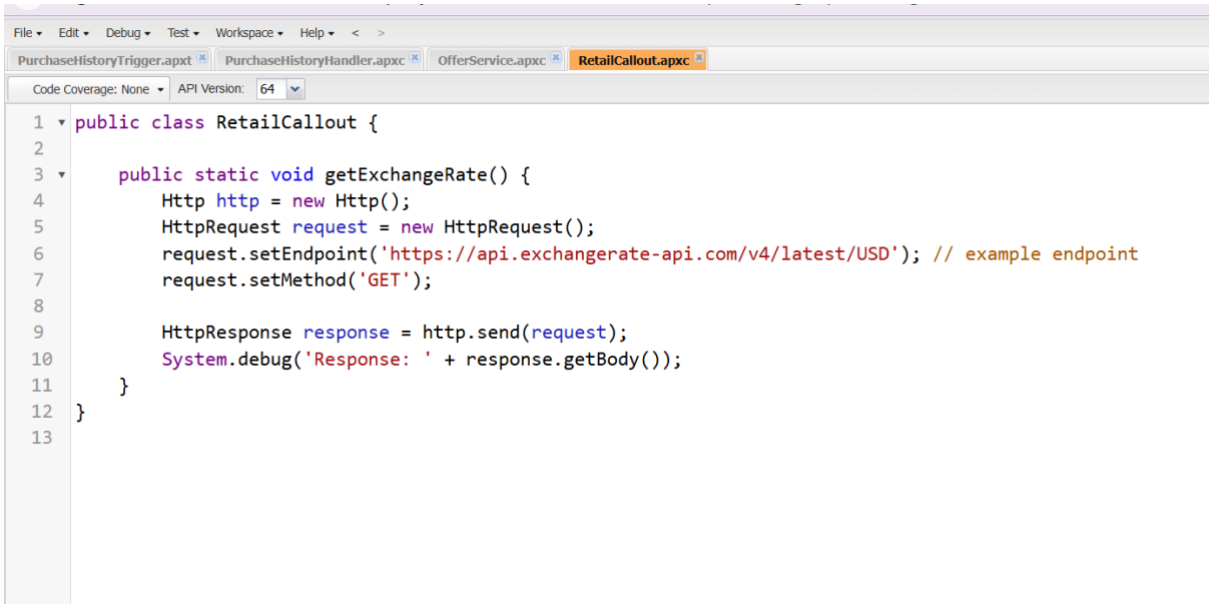
SOAP: Integrate with legacy ERP or payment systems that require SOAP for order processing or inventory updates.



Callouts

Use Case:

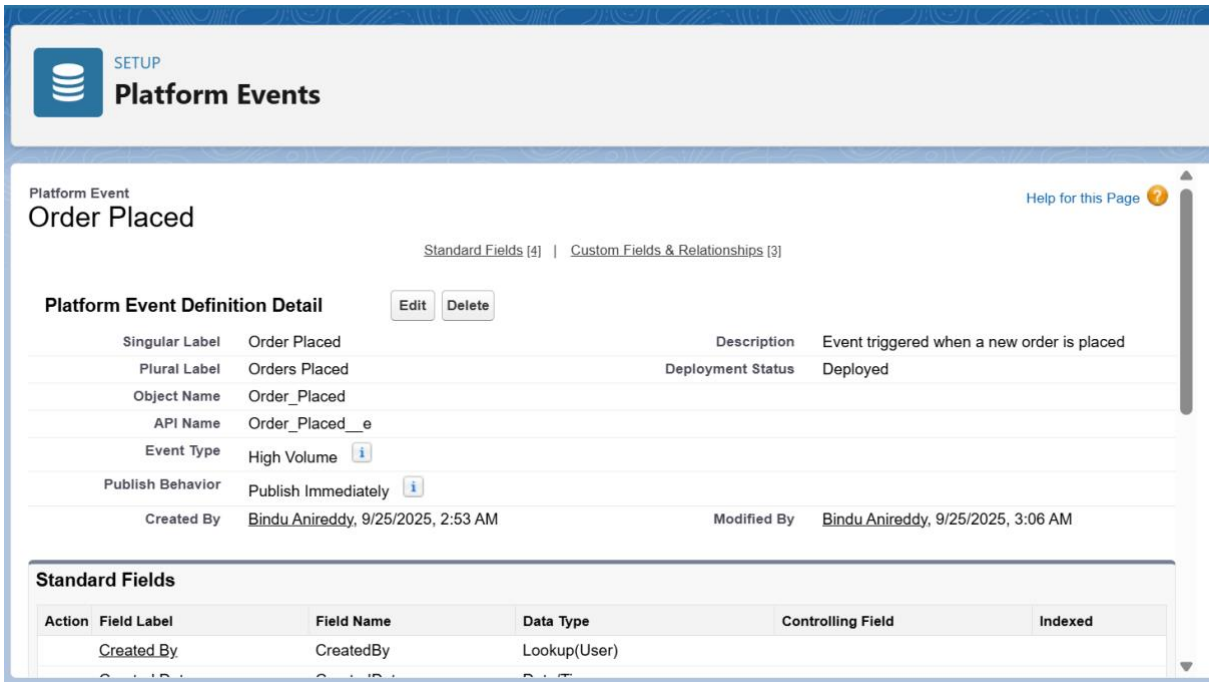
Fetch real-time **currency exchange rates, product prices, or stock levels** from external APIs to update Salesforce records automatically.



Platform Events

Use Case:

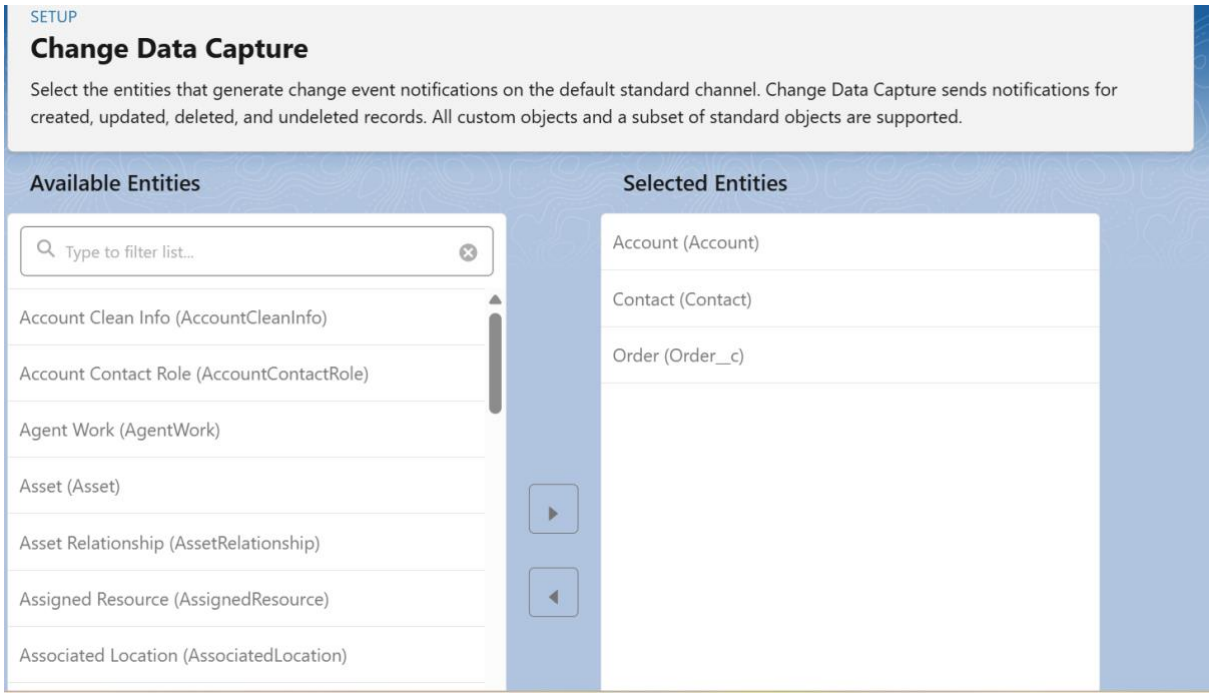
Notify other systems or processes in real-time when a new order is created, such as updating dashboards, sending emails, or triggering inventory updates.



Change Data Capture

Use Case:

Automatically capture changes in customer or order records to update dashboards, synchronize with external systems, or trigger automation flows in real-time.

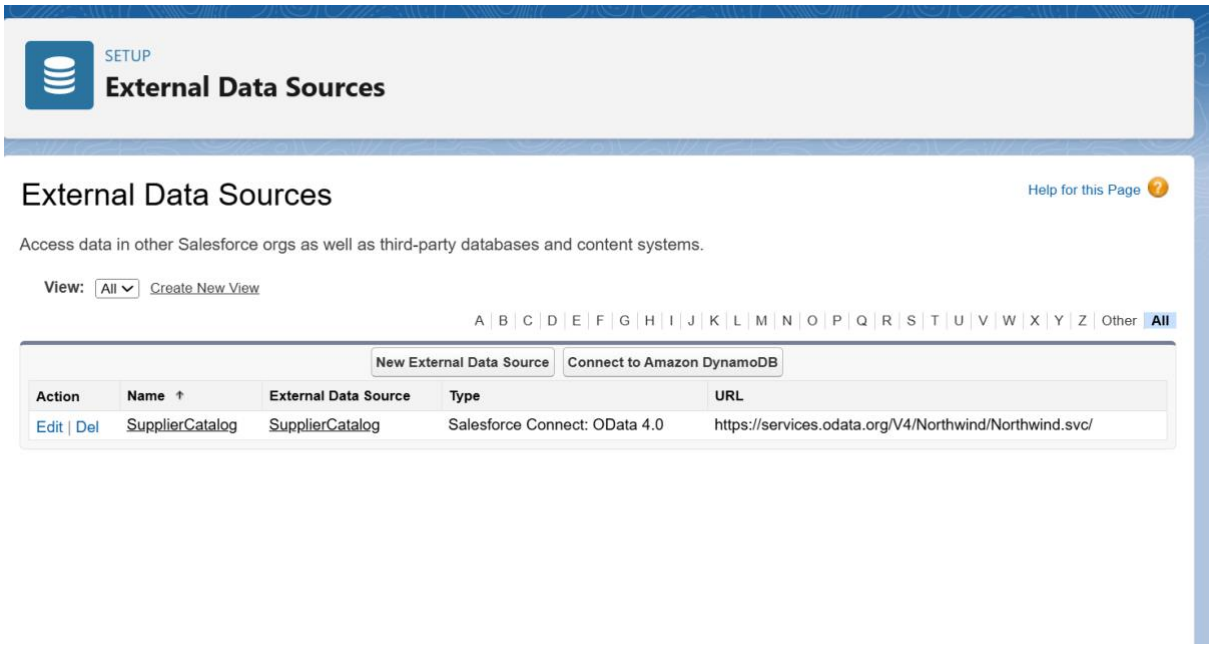


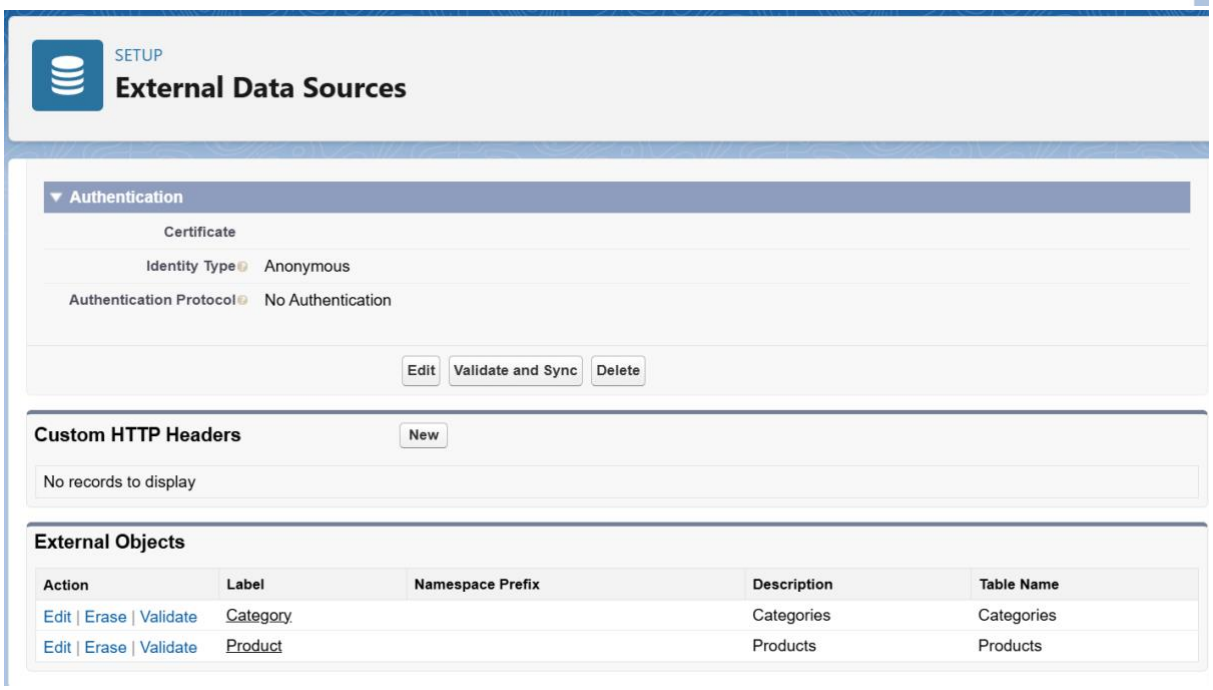
Salesforce Connect

Use Case:

Access and display **external system data**, like inventory or supplier info, in Salesforce without storing it locally.

Enables sales reps to **view and act on real-time external data** directly within Salesforce.



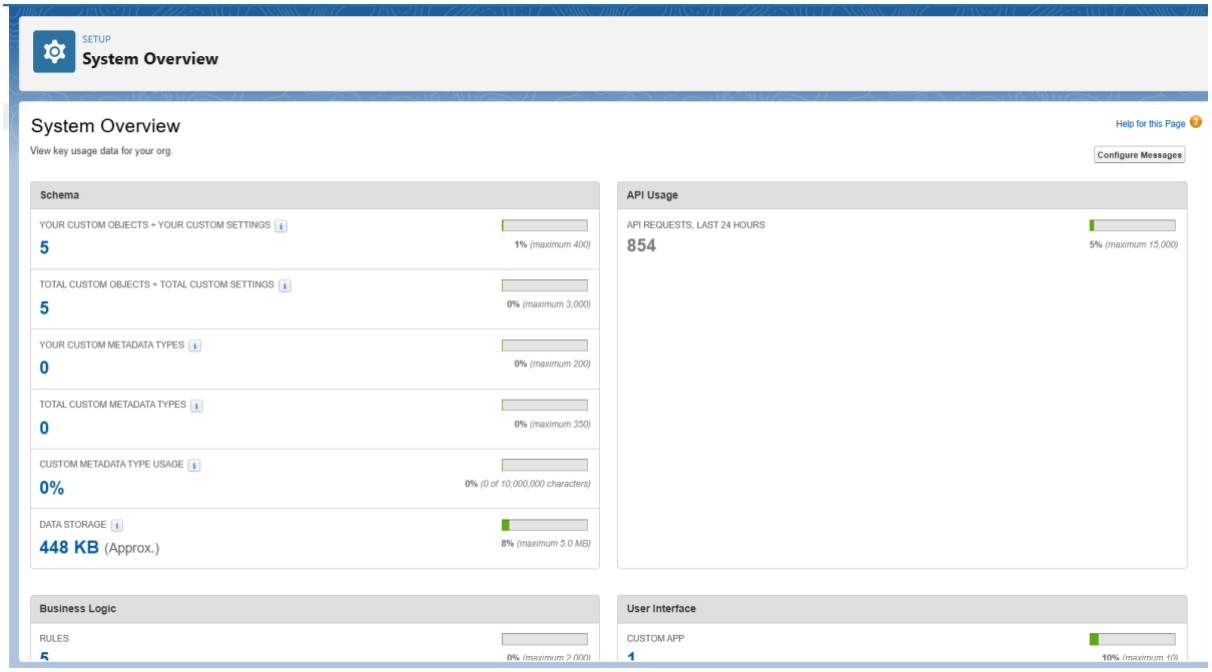


API Limits

Use Case:

Monitor and manage API usage to ensure integrations with external systems, like ERP or inventory platforms, **do not exceed Salesforce limits**.

Helps maintain **system performance and prevent failed data syncs** during high-volume retail operations.



OAuth & Authentication

Use Case:

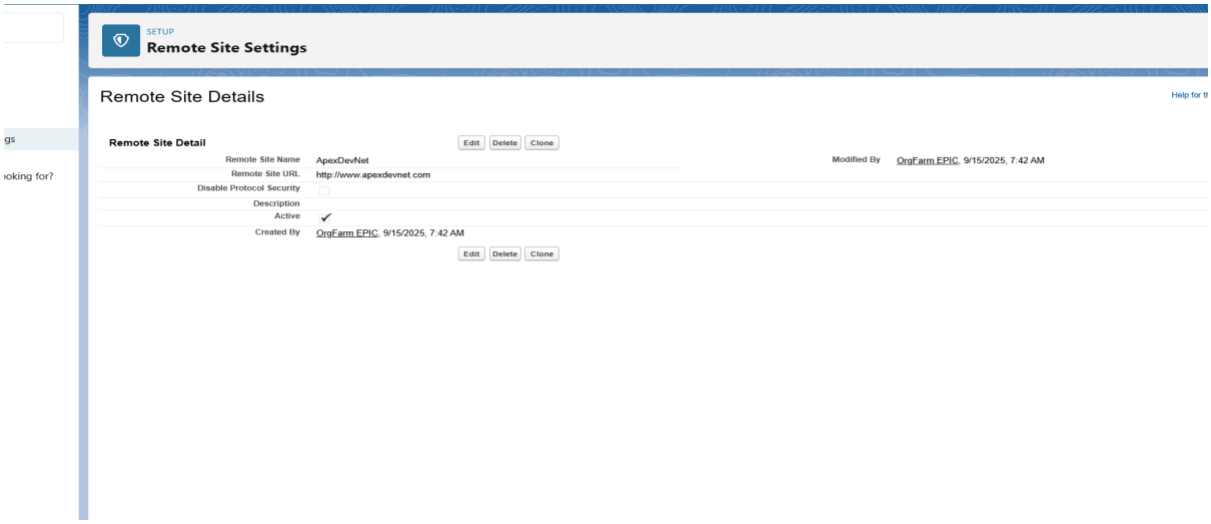
Enable secure integration between Salesforce and external systems, such as **mobile apps, ERP, or e-commerce platforms**, allowing real-time order updates and inventory synchronization

Remote Site Settings

Use Case:

Allow Salesforce to securely make **callouts to external systems**, such as payment gateways, supplier APIs, or shipping services.

Ensures that all HTTP requests to these external endpoints are **authorized**, preventing unauthorized access and protecting retail integrations.



Phase 8: Data Management & Deployment

Data Import Wizards

Use Case:

Quickly import **bulk Contact records** from CSV files into Salesforce without coding, saving time during customer onboarding.

Enables sales reps to have **all customer data available in Salesforce** for marketing, sales, and support activities.

Data Import Wizard

Help for this page

Recent Import Jobs

Status	Object	Records Created	Records Updated	Records Failed	Start Date	Processing Time (ms)
Closed	Contact	20	0	0	09-25-2025 12:12	404
Closed	Purchase History	0	0	20	09-25-2025 11:47	82
Closed	Purchase History	0	1	0	09-25-2025 11:41	284

Data Loader

Use Case:

Perform **bulk data operations** like inserting, updating, or deleting thousands of records (e.g., Orders, Products) efficiently.

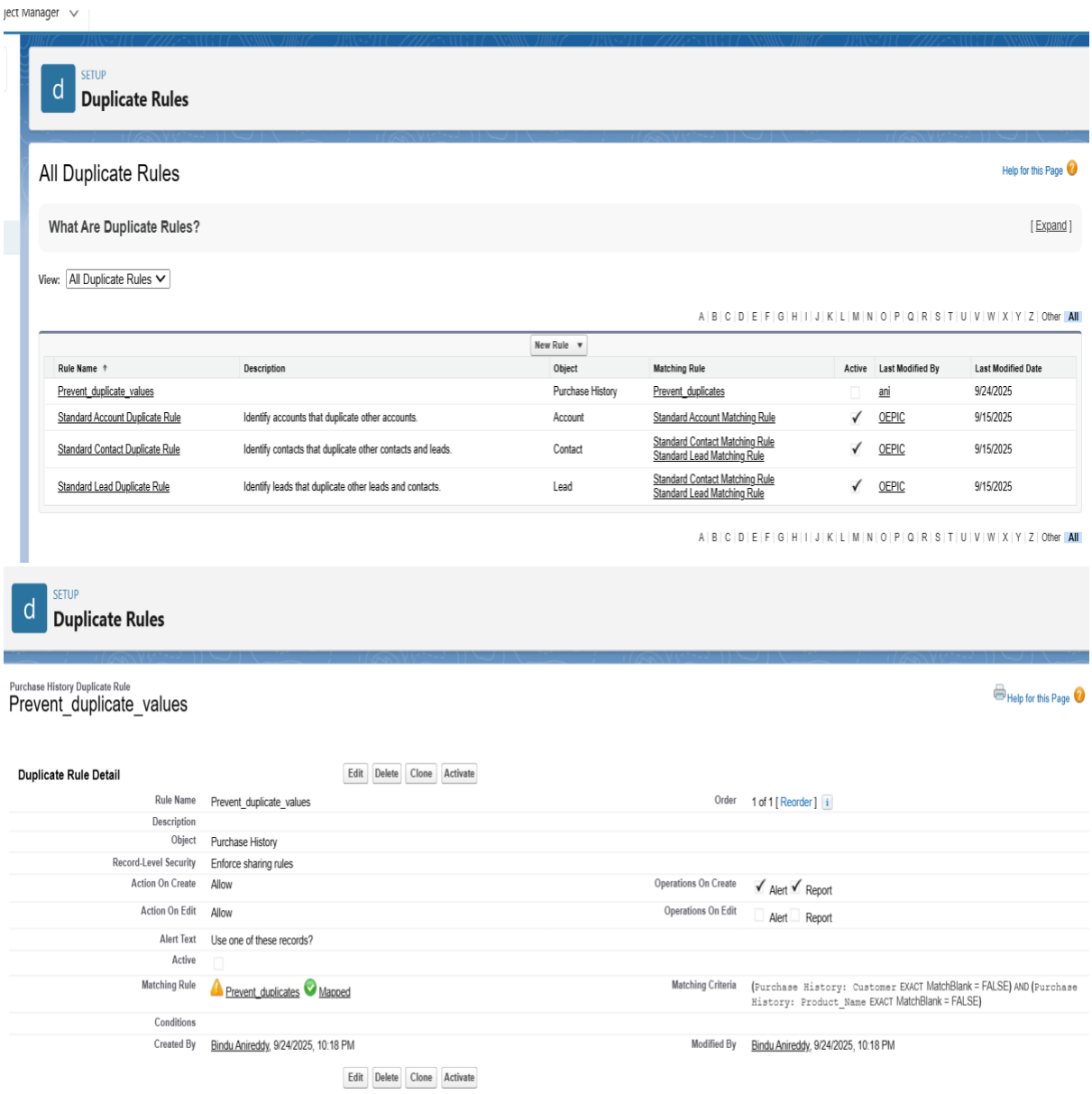
Ideal for **migrating large datasets** from legacy systems or external databases into Salesforce for retail operations.

Duplicate Rules

Use Case:

Prevent creation of **duplicate Contact or Customer records**, ensuring clean and accurate customer data.

Helps sales reps avoid contacting the same customer multiple times and **maintains data integrity** for reporting and analytics.

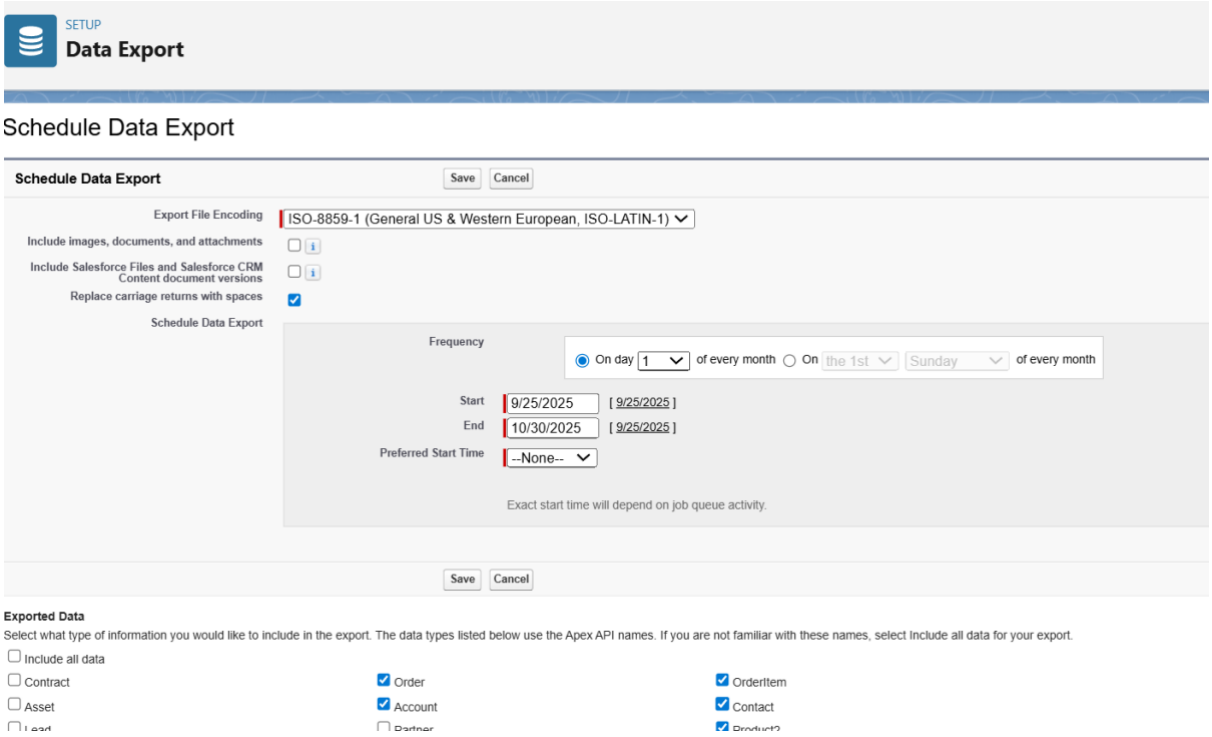


Data Export & Backup

Use Case:

Regularly back up **customer, order, and product data** to safeguard against accidental deletion or system failures.

Ensures **data recovery, compliance, and business continuity** for retail operations.



Change Sets

Use Case:

Move custom objects, LWCs, and Apex logic from sandbox to production efficiently, ensuring retail features like order processing and customer dashboards are deployed safely.

Unmanaged vs Managed Packages

Use Case:

- **Unmanaged Package:** Deploy **custom objects, fields, and components** from a sandbox to production for internal retail customizations; allows full editing in target org.
- **Managed Package:** Install **prebuilt third-party apps** (e.g., loyalty programs, payment integrations) in production; ensures controlled updates and protects intellectual property.

ANT Migration Tool

Use Case:

Automate deployment of large sets of metadata (objects, fields, Apex, LWCs) between Salesforce orgs, especially for complex retail projects.

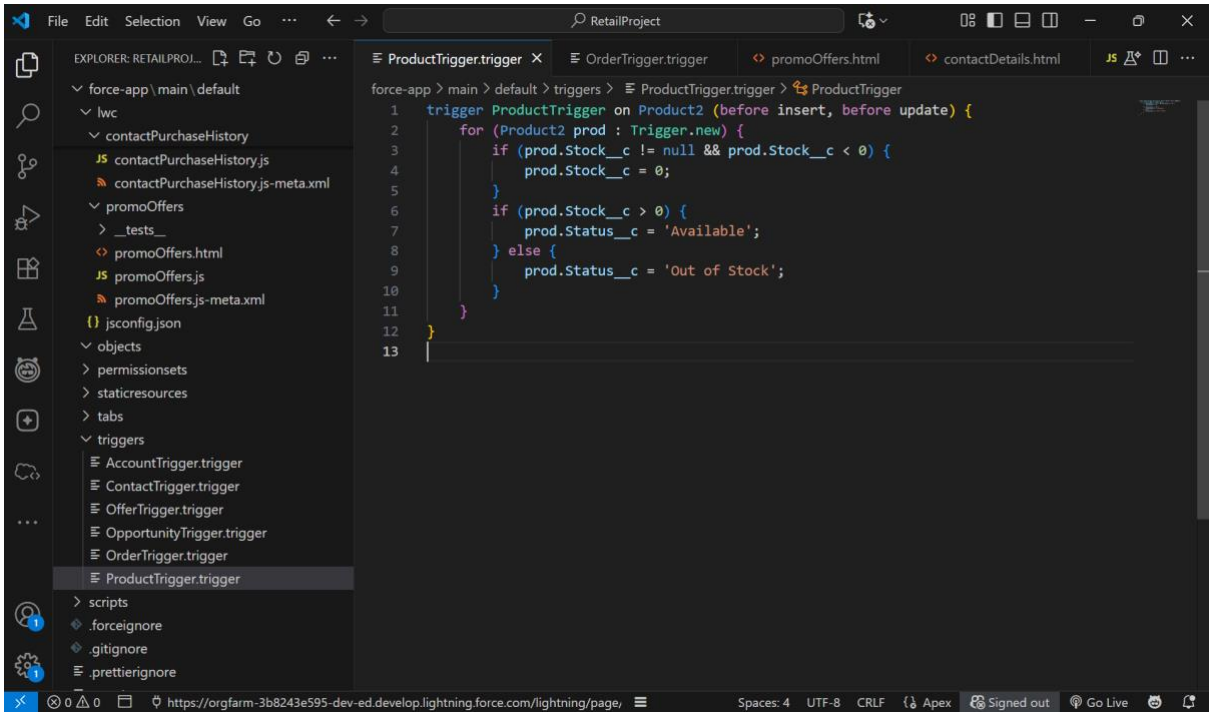
Ideal for version-controlled deployments and repetitive migrations from sandbox to production without using Change Sets.

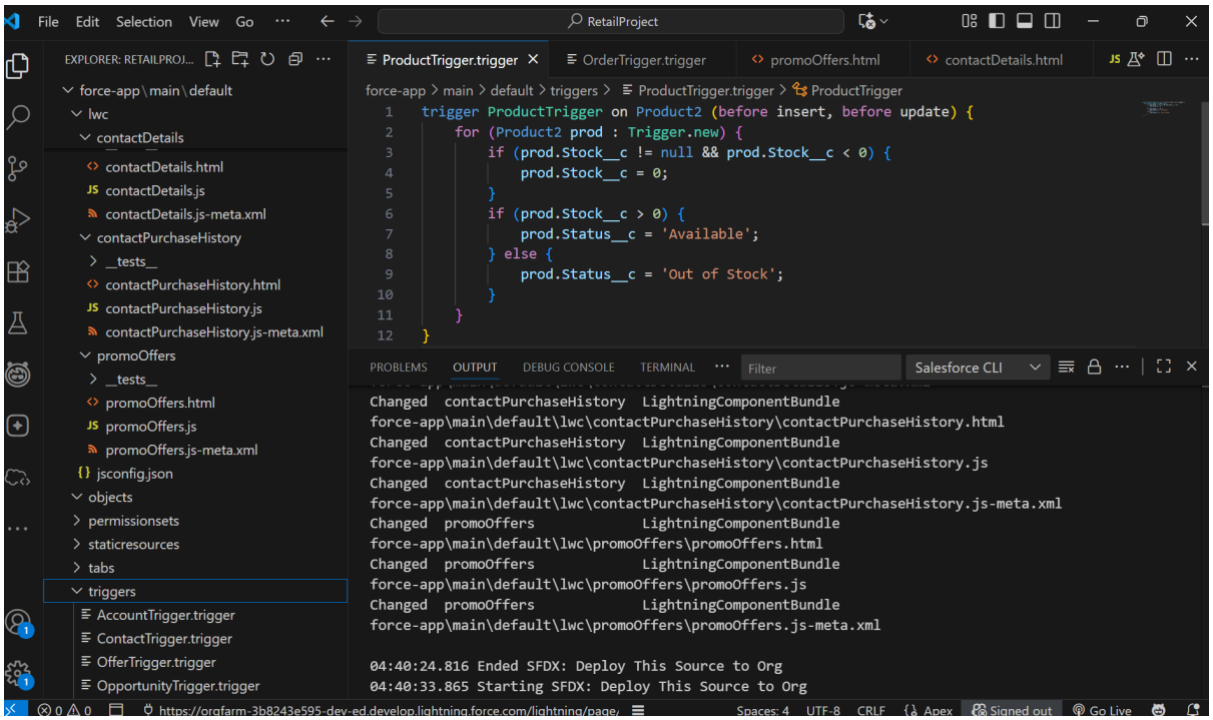
VS Code & SFDX

Use Case:

Develop, test, and deploy Salesforce metadata like Apex classes, LWCs, and Flows directly from VS Code using SFDX commands.

Enables source-driven development, version control, and rapid deployment for retail customizations and integrations.





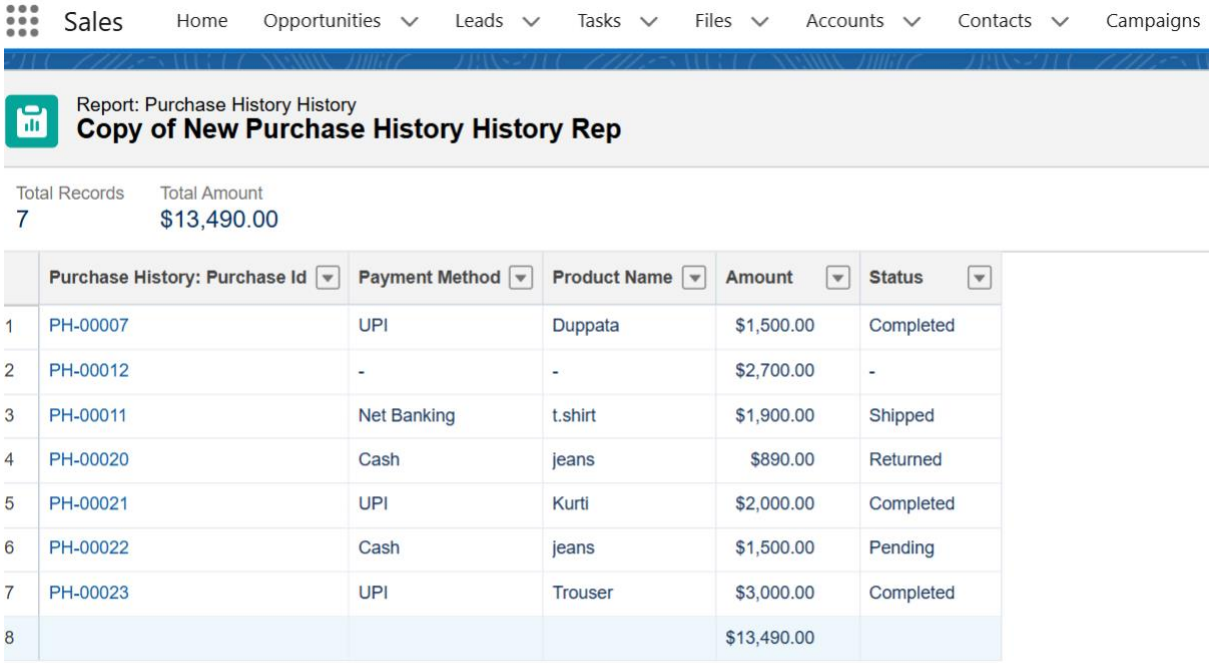
Phase 9: Reporting, Dashboards & Security Review

Reports (Tabular, Summary, Matrix, Joined)

Use Case:

Generate reports on Purchase History including Payment Method, Amount, and Status to monitor transactions.

Enable sales and finance teams to track payments, identify pending or failed transactions, and analyze sales trends for better retail decision-making.



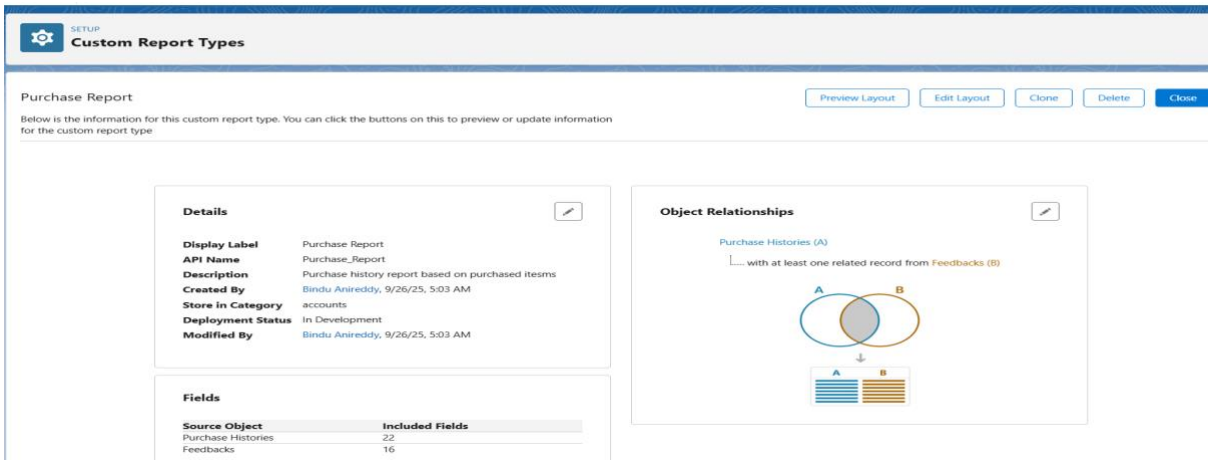
	Purchase History: Purchase Id	Payment Method	Product Name	Amount	Status
1	PH-00007	UPI	Duppata	$1,500.00	Completed
2	PH-00012	-	-	$2,700.00	-
3	PH-00011	Net Banking	t.shirt	$1,900.00	Shipped
4	PH-00020	Cash	jeans	$890.00	Returned
5	PH-00021	UPI	Kurti	$2,000.00	Completed
6	PH-00022	Cash	jeans	$1,500.00	Pending
7	PH-00023	UPI	Trouser	$3,000.00	Completed
8				$13,490.00	

Report Types

Use Case:

Create custom report types combining Purchase History and Customer Feedback to analyze which products or orders received positive or negative feedback.

Helps sales and product teams correlate purchases with customer satisfaction, enabling targeted improvements in products, service, and promotions.

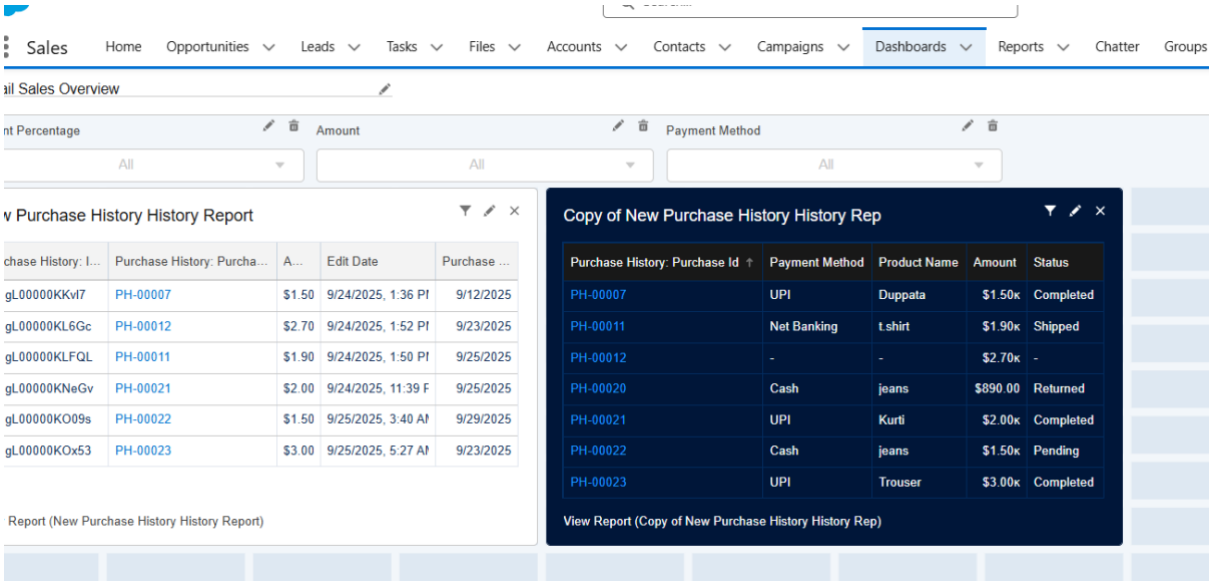


Dashboards

Use Case:

Visualize key retail metrics like **total sales, top-selling products, pending orders, and customer satisfaction** in one place.

Enables managers and sales reps to **quickly track performance, identify trends, and make informed decisions** without running multiple reports.

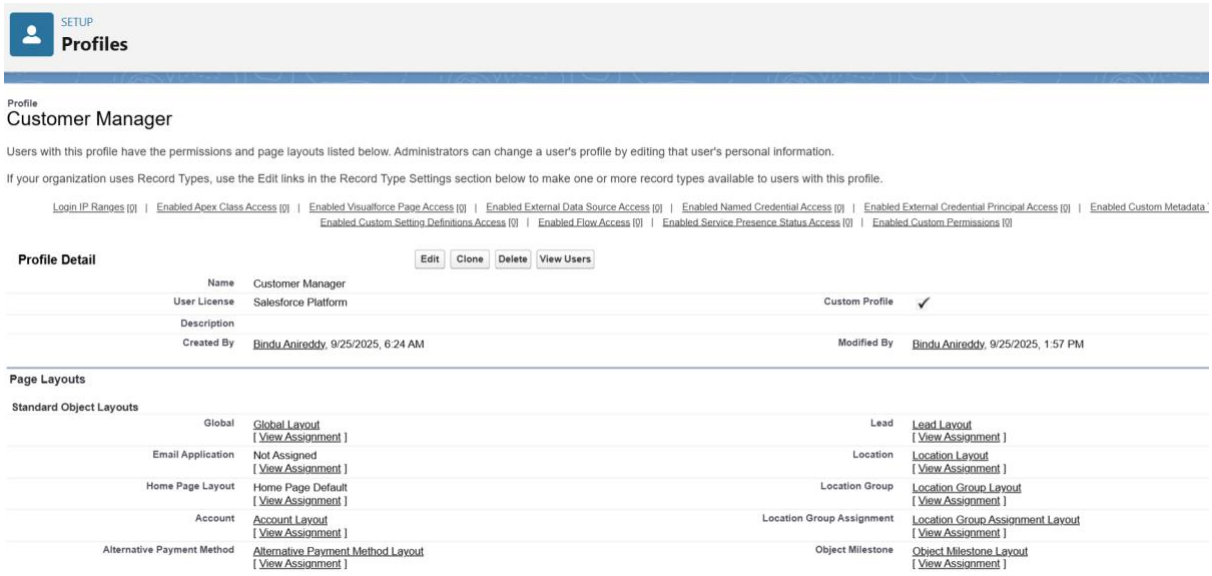


Profiles

Use Case:

Control **what users can see and do** in Salesforce by assigning profiles with specific object, field, and app permissions.

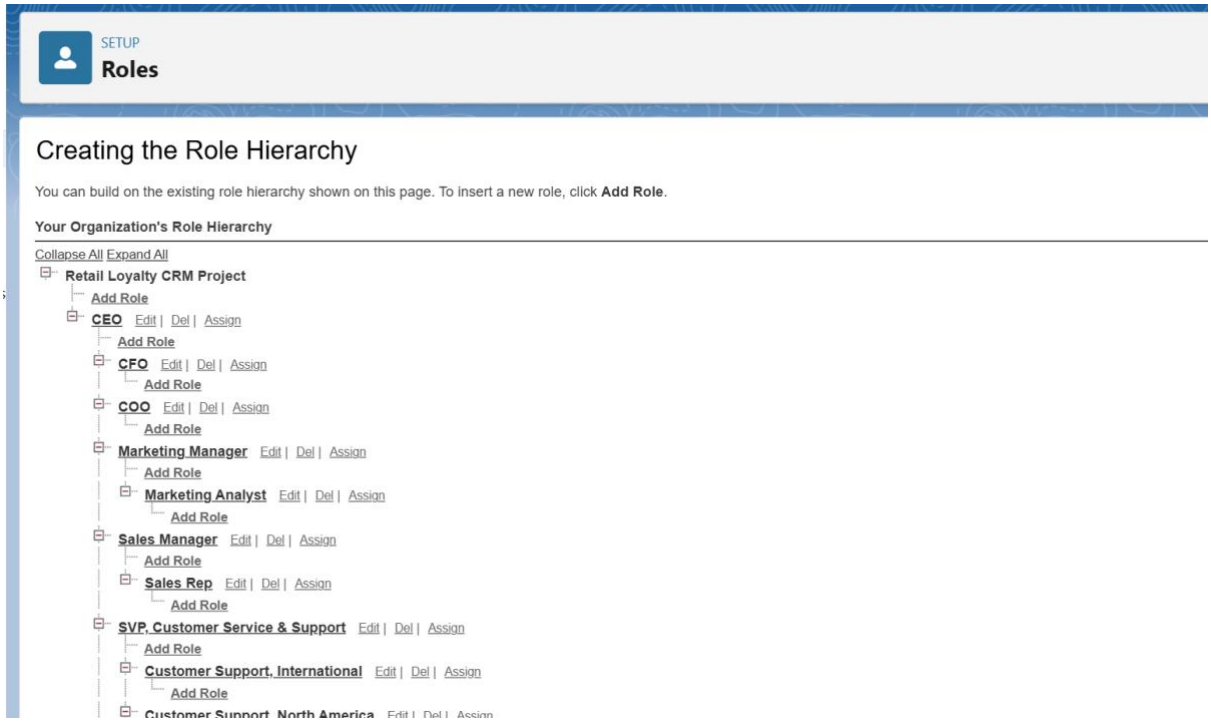
For example, a **Sales Rep profile** can access Purchase History and Accounts but not sensitive financial settings, ensuring secure and role-based access in retail operations.



Roles

Use Case:

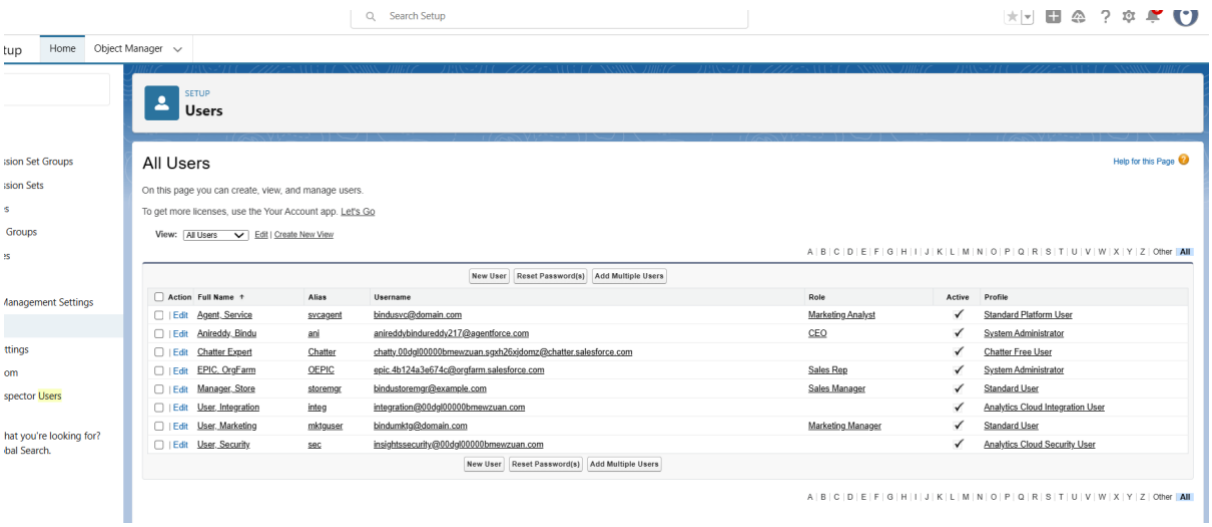
Define a **hierarchy of users** to control record visibility and sharing.
For example, a **Store Manager role** can see all orders from their team, while **Sales Reps** only see their own, enabling secure and organized access to retail data.



Users

Use Case:

Create and manage **individual accounts** for employees.
Example: Add a **new cashier** as a Salesforce User with limited retail app access.

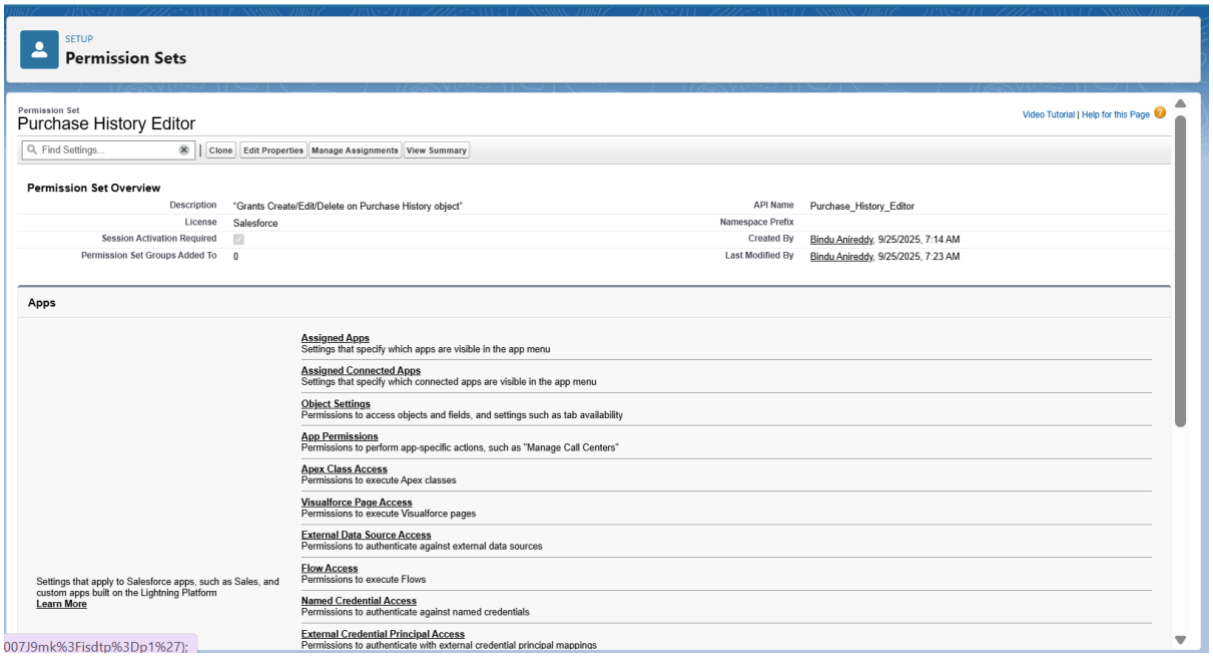


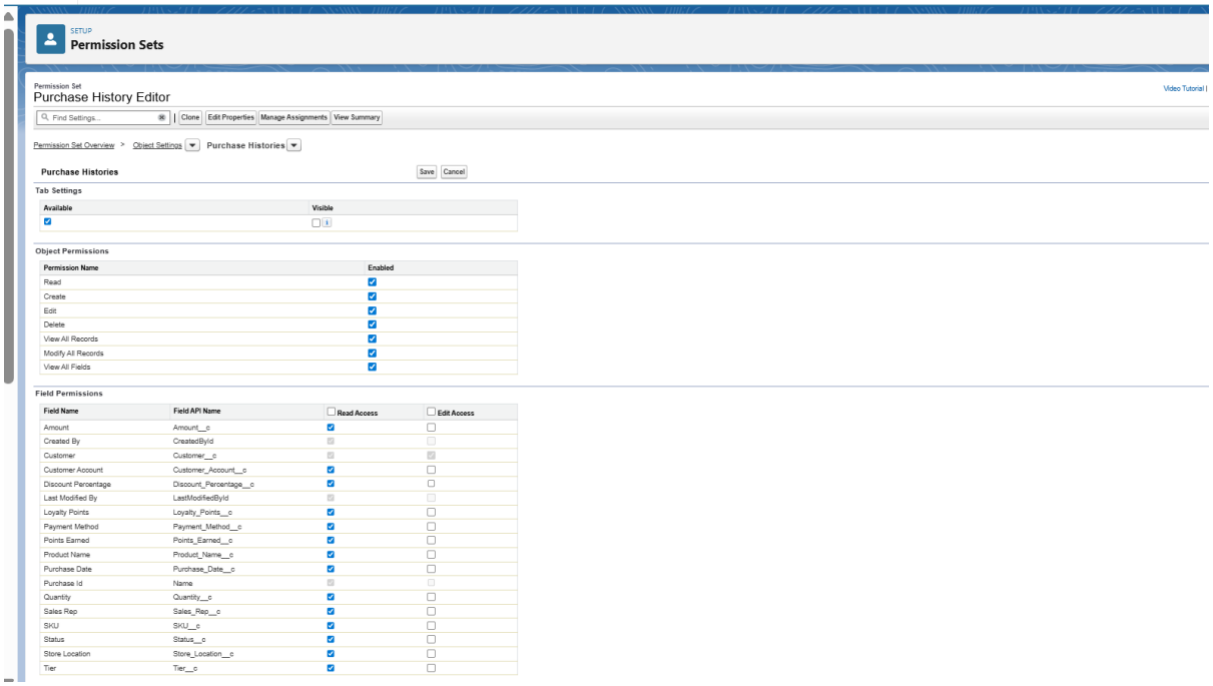
Permission sets

Use Case:

Grant additional permissions to specific users without changing their profile.

Example: Give a junior sales rep access to a custom dashboard temporarily**.**



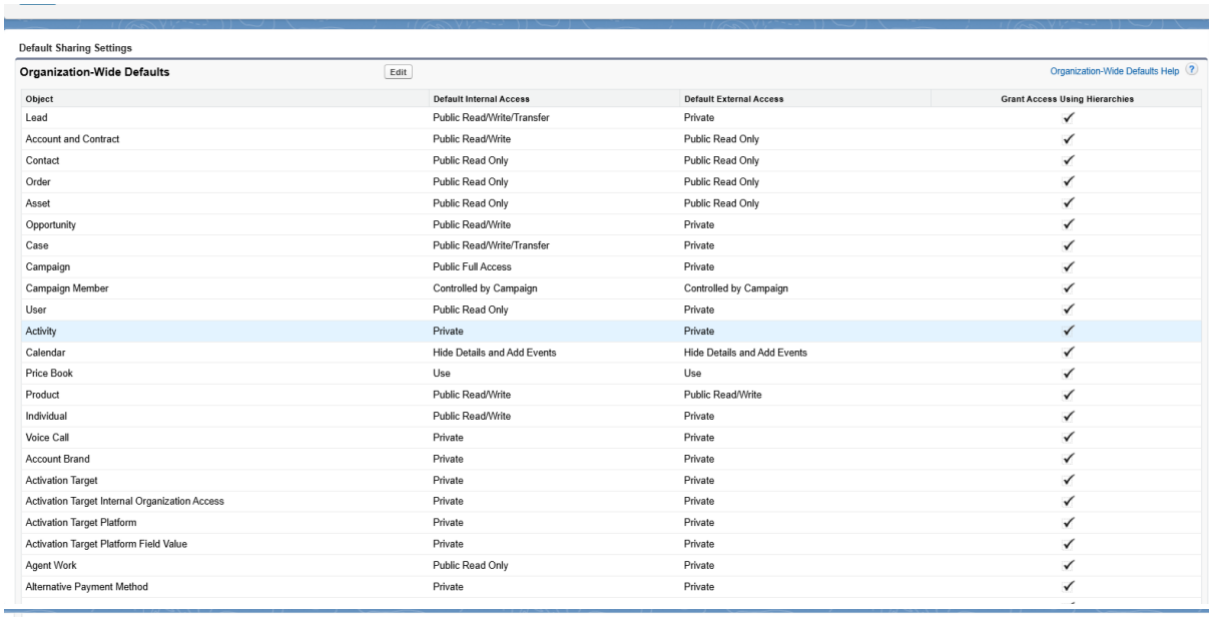


OWD

Use Case:

Set **baseline record visibility** across the org.

Example: Make Order__c records **private** so sales reps only see their own orders.



Default Sharing Settings

Organization-Wide Defaults Edit — Organization-Wide Defaults Help

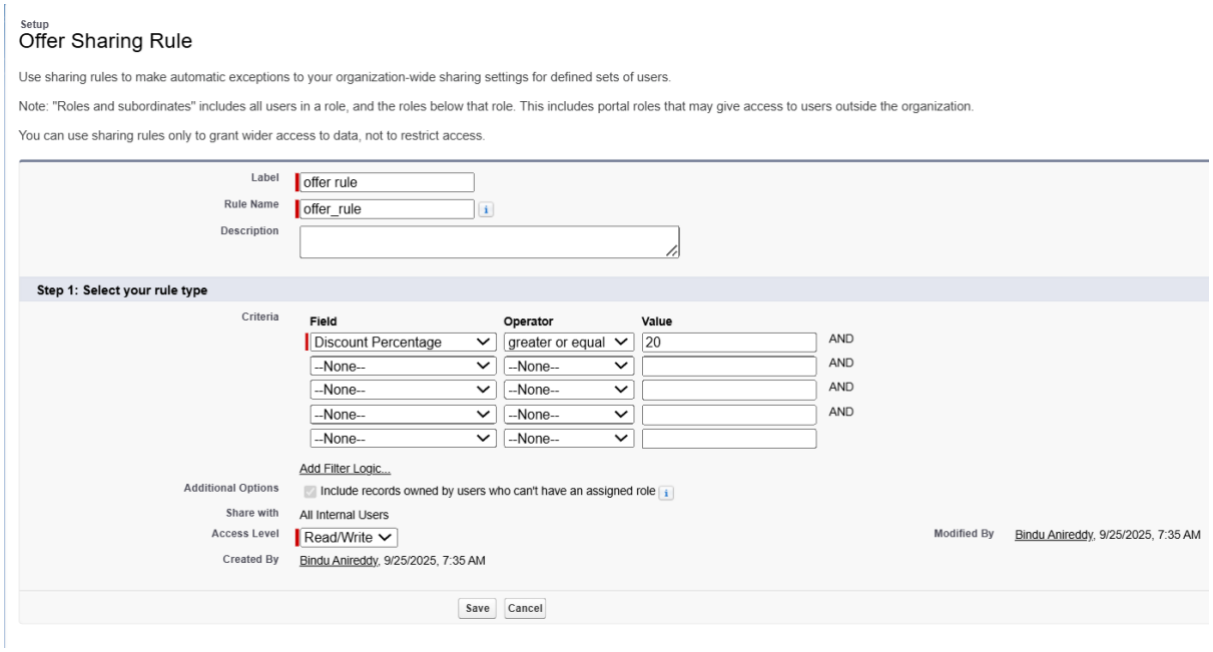
Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	Private	✓
Account and Contract	Public Read/Write	Public Read Only	✓
Contact	Public Read Only	Public Read Only	✓
Order	Public Read Only	Public Read Only	✓
Asset	Public Read Only	Public Read Only	✓
Opportunity	Public Read/Write	Private	✓
Case	Public Read/Write/Transfer	Private	✓
Campaign	Public Full Access	Private	✓
Campaign Member	Controlled by Campaign	Controlled by Campaign	✓
User	Public Read Only	Private	✓
Activity	Private	Private	✓
Calendar	Hide Details and Add Events	Hide Details and Add Events	✓
Price Book	Use	Use	✓
Product	Public Read/Write	Public Read/Write	✓
Individual	Public Read/Write	Private	✓
Voice Call	Private	Private	✓
Account Brand	Private	Private	✓
Activation Target	Private	Private	✓
Activation Target Internal Organization Access	Private	Private	✓
Activation Target Platform	Private	Private	✓
Activation Target Platform Field Value	Private	Private	✓
Agent Work	Public Read Only	Private	✓
Alternative Payment Method	Private	Private	✓
Service Territory	Public Read/Write	Private	
Shift	Private	Private	
Shipment	Private	Private	
Shipping Carrier	Public Read Only	Private	
Shipping Carrier Method	Public Read Only	Private	
Shipping Configuration Set	Public Read Only	Private	
Streaming Channel	Public Read/Write	Private	
Tableau Host Mapping	Public Read Only	Private	
User Presence	Public Read Only	Private	
User Provisioning Request	Private	Private	
Waitlist	Private	Private	
Web Cart Document	Private	Private	
Work Order	Private	Private	
Work Plan	Private	Private	
Work Plan Template	Private	Private	
Work Step Template	Private	Private	
Work Type	Private	Private	
Work Type Group	Public Read/Write	Private	
Contact Offer	Controlled by Parent	Controlled by Parent	
Feedback	Controlled by Parent	Controlled by Parent	
Offer	Public Read/Write	Private	
Order	Public Read/Write	Private	
Purchase History	Controlled by Parent	Controlled by Parent	

Sharing Rules

Use Case:

Automatically **grant record access** based on criteria or ownership.

Example: Share all **high-value orders** with store managers across teams.

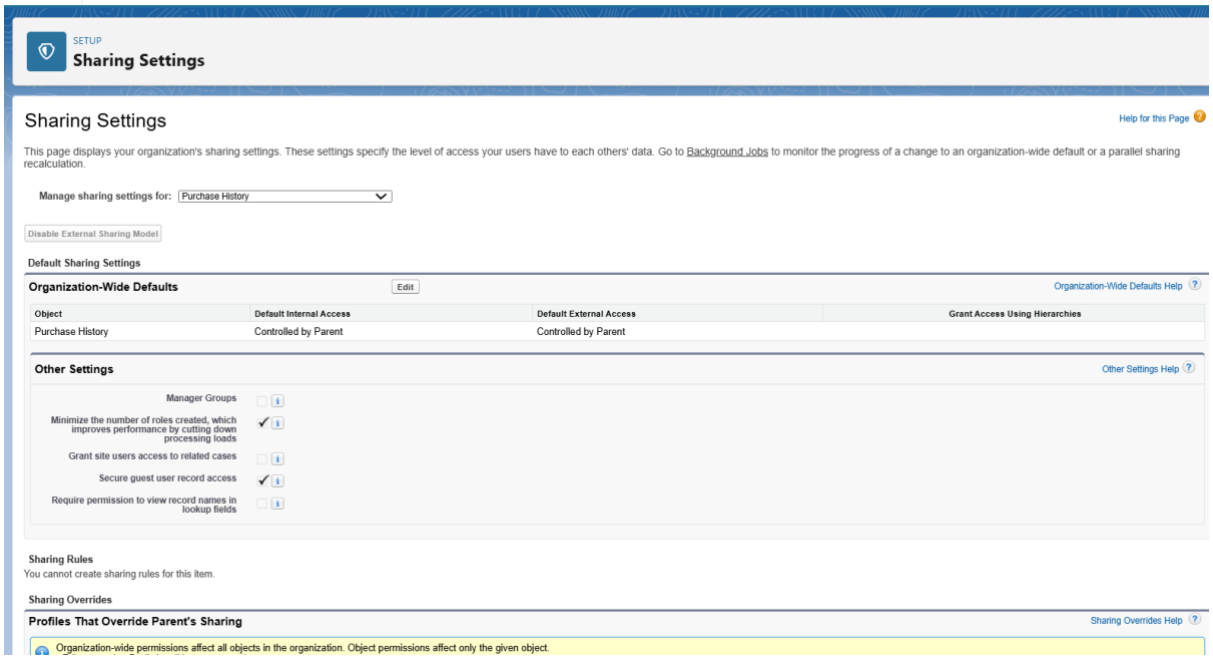


Sharing settings

Use Case:

Define overall sharing model combining OWD, roles, and sharing rules.

Example: Control whether Customer Feedback is private, read-only, or public within the retail org.

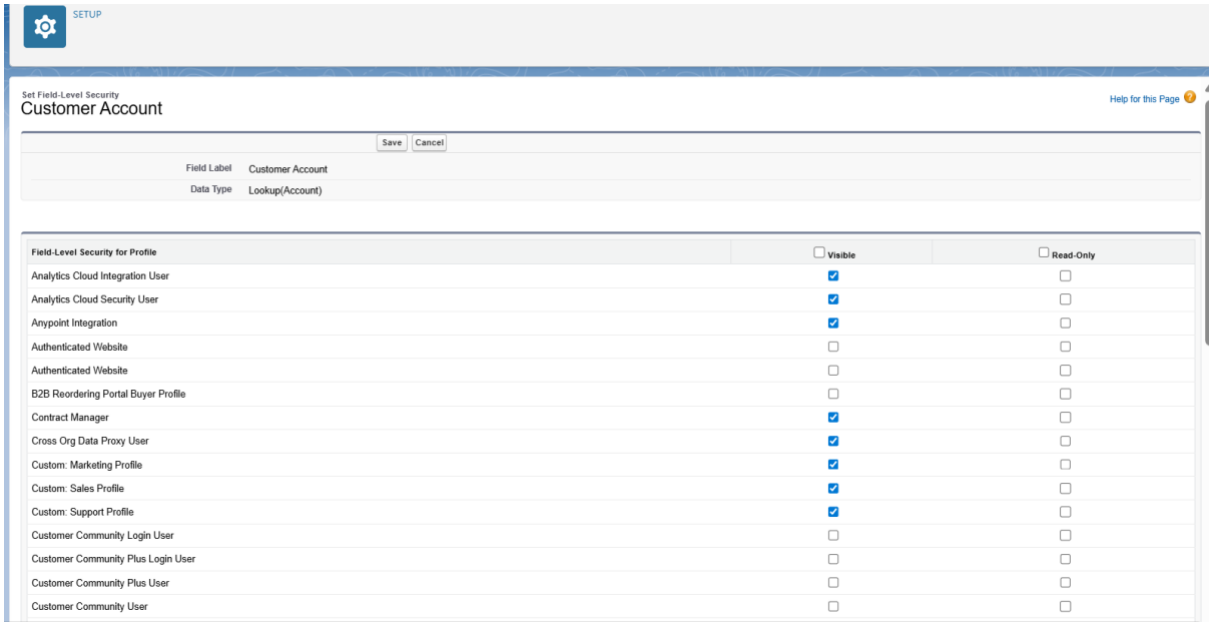


Field Level Security

Use Case:

Control **visibility and edit rights** for individual fields on objects.

Example: Hide Discount Percentage from junior sales reps but allow managers to edit.

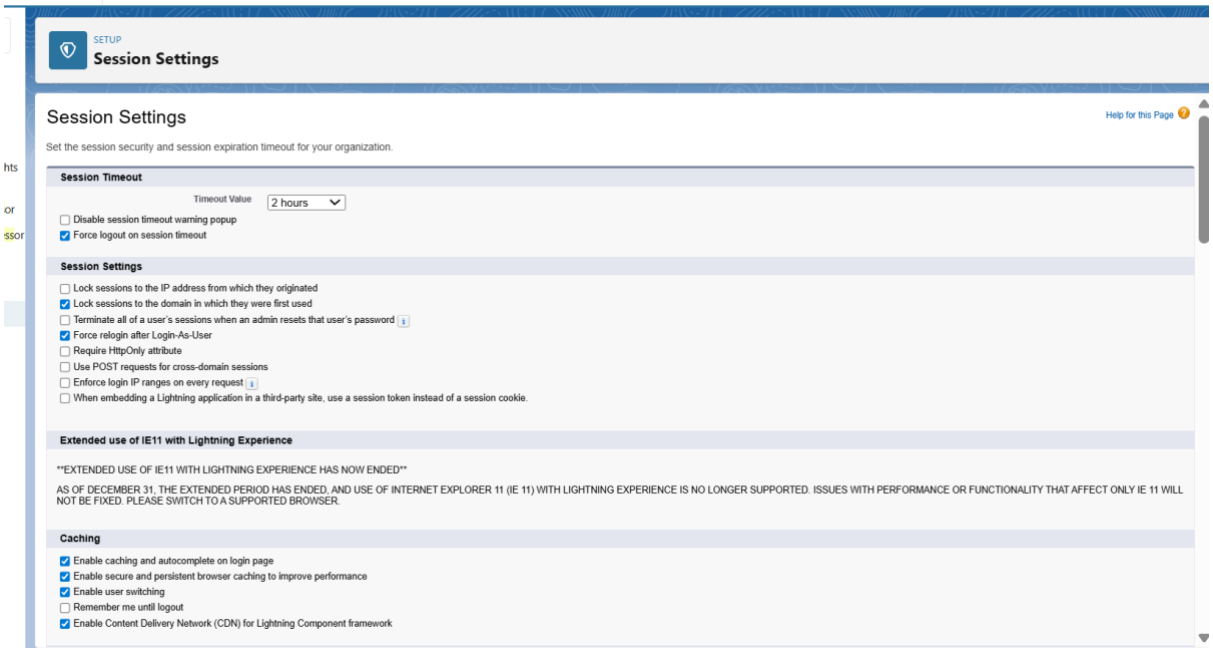


Session Settings

Use Case:

Configure **session security, timeout, and login policies**.

Example: Automatically log out users after 30 minutes of inactivity to protect sensitive retail data.



Login IP Ranges

Use Case:

Restrict user logins to specific IP addresses.

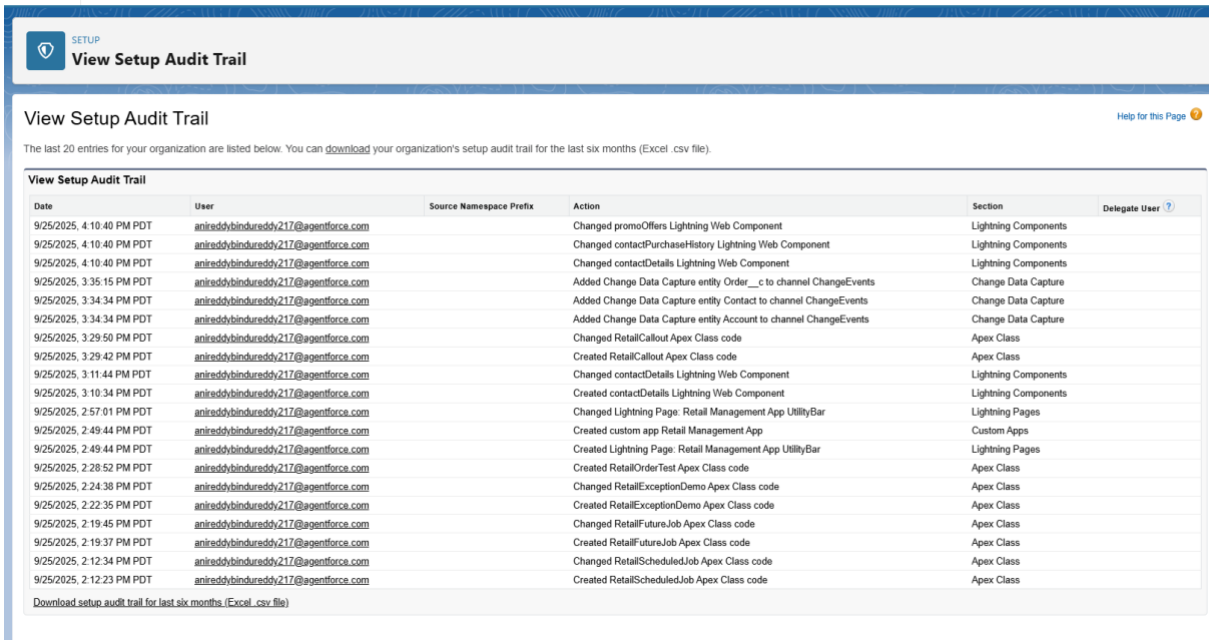
Example: Allow sales reps to log in only from store or corporate office networks, preventing unauthorized access**.**

Audit Trail

Use Case:

Track **setup changes and administrative actions** for security and compliance.

Example: Monitor who modified **discount rules or product pricing** in the retail org over the last 6 months.



SETUP
View Setup Audit Trail

View Setup Audit Trail

Help for this Page

The last 20 entries for your organization are listed below. You can download your organization's setup audit trail for the last six months (Excel .csv file).

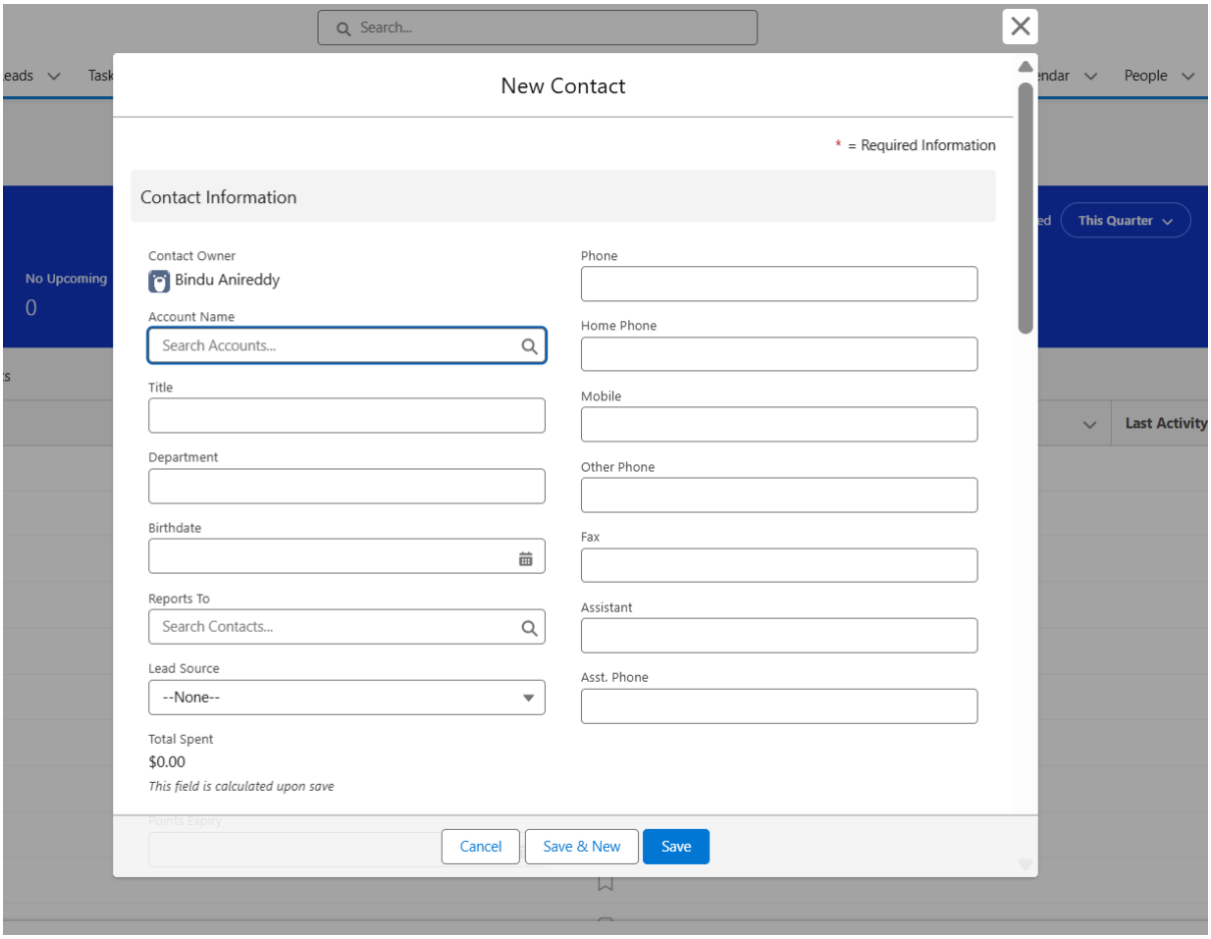
View Setup Audit Trail

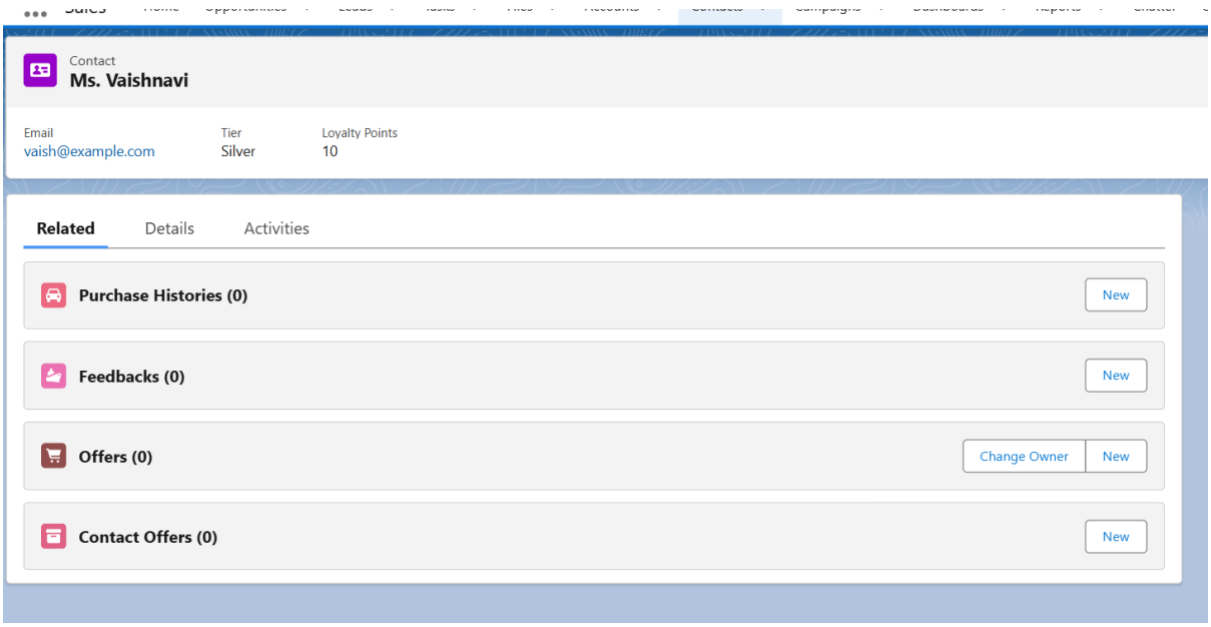
Date	User	Source Namespace Prefix	Action	Section	Delegate User
9/25/2025, 4:10:40 PM PDT	anireddybindureddy217@agentforce.com		Changed promoOffers Lightning Web Component	Lightning Components	
9/25/2025, 4:10:40 PM PDT	anireddybindureddy217@agentforce.com		Changed contactPurchaseHistory Lightning Web Component	Lightning Components	
9/25/2025, 4:10:40 PM PDT	anireddybindureddy217@agentforce.com		Changed contactDetails Lightning Web Component	Lightning Components	
9/25/2025, 3:35:15 PM PDT	anireddybindureddy217@agentforce.com		Added Change Data Capture entity Order__c to channel ChangeEvents	Change Data Capture	
9/25/2025, 3:34:34 PM PDT	anireddybindureddy217@agentforce.com		Added Change Data Capture entity Contact to channel ChangeEvents	Change Data Capture	
9/25/2025, 3:34:34 PM PDT	anireddybindureddy217@agentforce.com		Added Change Data Capture entity Account to channel ChangeEvents	Change Data Capture	
9/25/2025, 3:29:50 PM PDT	anireddybindureddy217@agentforce.com		Changed RetailCallout Apex Class code	Apex Class	
9/25/2025, 3:29:42 PM PDT	anireddybindureddy217@agentforce.com		Created RetailCallout Apex Class code	Apex Class	
9/25/2025, 3:11:44 PM PDT	anireddybindureddy217@agentforce.com		Changed contactDetails Lightning Web Component	Lightning Components	
9/25/2025, 3:10:34 PM PDT	anireddybindureddy217@agentforce.com		Created contactDetails Lightning Web Component	Lightning Components	
9/25/2025, 2:57:01 PM PDT	anireddybindureddy217@agentforce.com		Changed Lightning Page: Retail Management App UtilityBar	Lightning Pages	
9/25/2025, 2:49:44 PM PDT	anireddybindureddy217@agentforce.com		Created custom app Retail Management App	Custom Apps	
9/25/2025, 2:49:44 PM PDT	anireddybindureddy217@agentforce.com		Created Lightning Page: Retail Management App UtilityBar	Lightning Pages	
9/25/2025, 2:28:52 PM PDT	anireddybindureddy217@agentforce.com		Created RetailOrderTest Apex Class code	Apex Class	
9/25/2025, 2:24:38 PM PDT	anireddybindureddy217@agentforce.com		Changed RetailExceptionDemo Apex Class code	Apex Class	
9/25/2025, 2:22:35 PM PDT	anireddybindureddy217@agentforce.com		Created RetailExceptionDemo Apex Class code	Apex Class	
9/25/2025, 2:19:45 PM PDT	anireddybindureddy217@agentforce.com		Changed RetailFutureJob Apex Class code	Apex Class	
9/25/2025, 2:19:37 PM PDT	anireddybindureddy217@agentforce.com		Created RetailFutureJob Apex Class code	Apex Class	
9/25/2025, 2:12:34 PM PDT	anireddybindureddy217@agentforce.com		Changed RetailScheduledJob Apex Class code	Apex Class	
9/25/2025, 2:12:23 PM PDT	anireddybindureddy217@agentforce.com		Created RetailScheduledJob Apex Class code	Apex Class	

Download setup audit trail for last six months (Excel .csv file)

Phase 10: Quality Assurance Testing

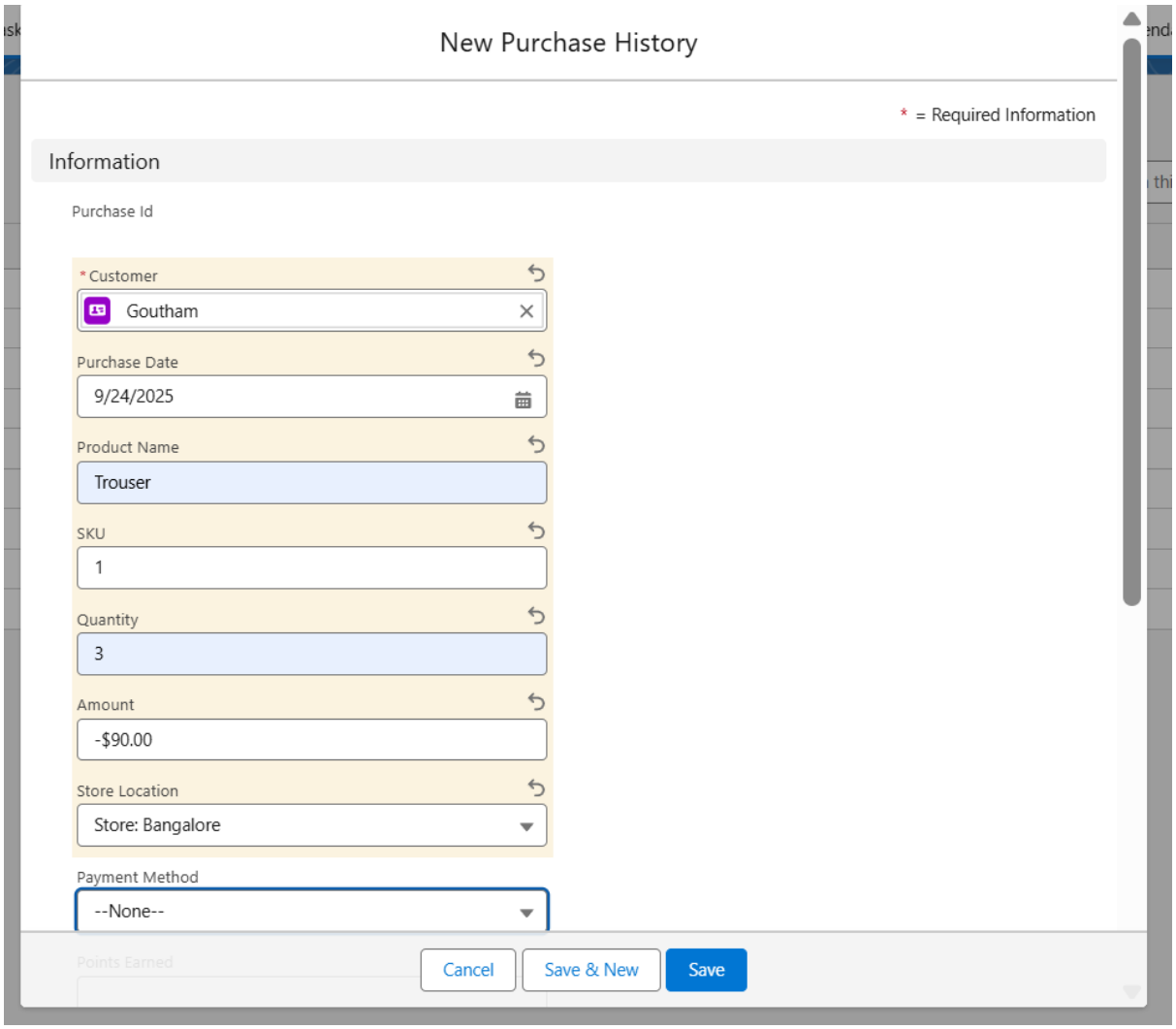
- **Test Case: Customer Registration**
- **Use Case / Scenario:** Create a new customer record in Customer__c
- **Test Steps (Input):**
  - Go to Contacts tab → Click New
  - Enter:
  - Name: Vaishnavi
  - Email: vaish@example.com
  - Phone:987654321
  - Loyalty points: 10
  - Tier:Silver
- **Expected Result**: A new Customer record is created with all details saved.
- **Actual Result**: Customer record created successfully

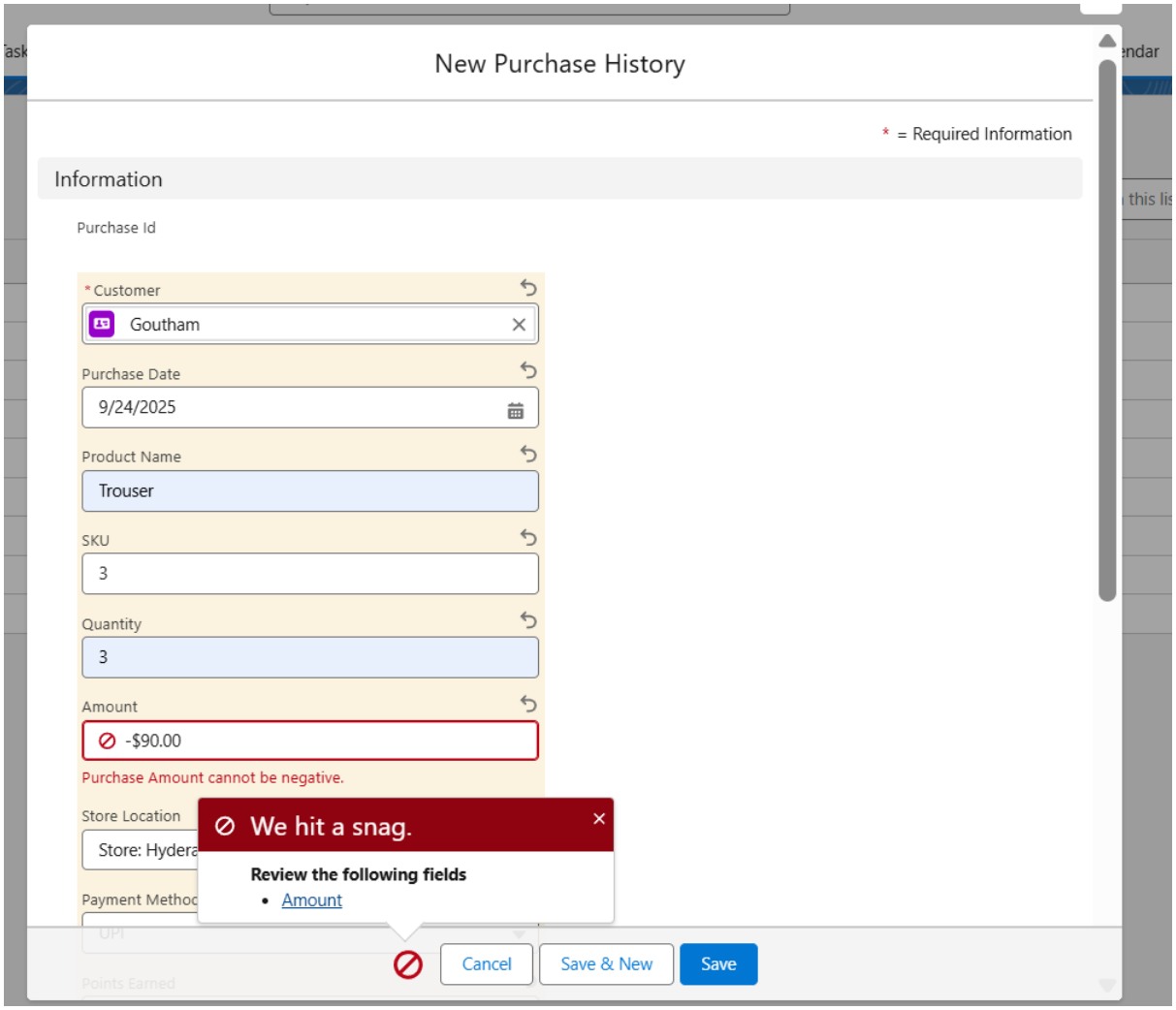

Search...
New Contact
* = Required Information
Contact Information
Contact Owner
Bindu Anireddy
Account Name
Search Accounts...
Title
Department
Birthdate
Reports To
Search Contacts...
Lead Source
--None--
Total Spent
$0.00
This field is calculated upon save
Phone
Home Phone
Mobile
Other Phone
Fax
Assistant
Asst. Phone
Cancel
Save & New
Save
This Quarter
No Upcoming
0
Last Activity
People


Contact
Ms. Vaishnavi
Email
vaish@example.com
Tier
Silver
Loyalty Points
10
Related
Details
Activities
Purchase Histories (0)
New
Feedbacks (0)
New
Offers (0)
Change Owner
New
Contact Offers (0)
New

Test Case: Prevent Negative Purchase Amount

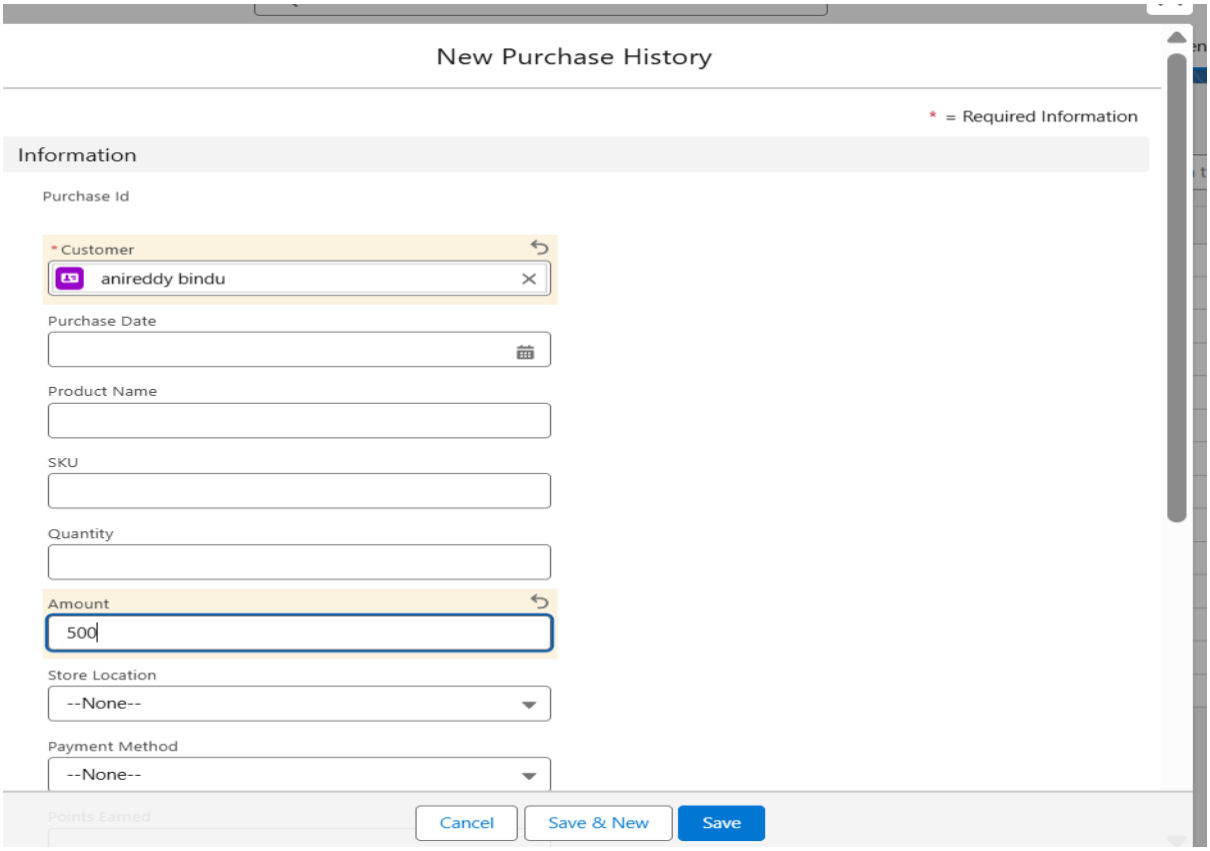
- **Use Case / Scenario:** Ensure that users cannot enter a negative purchase amount in Purchase History.
- **Test Steps (Input):**
    1. Go to Purchase History tab → Click New.
    2. Enter details:
        - Customer: Goutham
        - Purchase Date: 24-Sep-2025
        - Purchase Amount: -90
    3. Save the record.
- **Expected Result:**
    - System should prevent saving the record.
    - Error message should appear: "Purchase Amount cannot be Negative"
- **Actual Result:**
    - Record was not saved.
    - Validation error message appeared.





Test Case: Loyalty Points Auto-Update (Trigger/Flow)

- **Use Case / Scenario:** Adding a Purchase should update Customer's Loyalty Balance.
- **Test Steps (Input):**
    1. Create a new Purchase:
        - Customer: Anireddy Bindu
        - Purchase Amount: 500
    2. Save the record.
- **Expected Result:**
    - A new Loyalty Points record is created automatically.
    - Customer's Loyalty Balance increases by 50 points (if 1 point = ₹100).
- **Actual Result:** Loyalty updated correctly in Customer record.



Purchase History
PH-00030

Purchase Id
PH-00030

Customer
anireddy bindu

Purchase Date
9/26/2025

Product Name
Kurti

SKU

Quantity
2

Amount
$700.00

Store Location
Store: Hyderabad

Payment Method
Cash

Points Earned
70

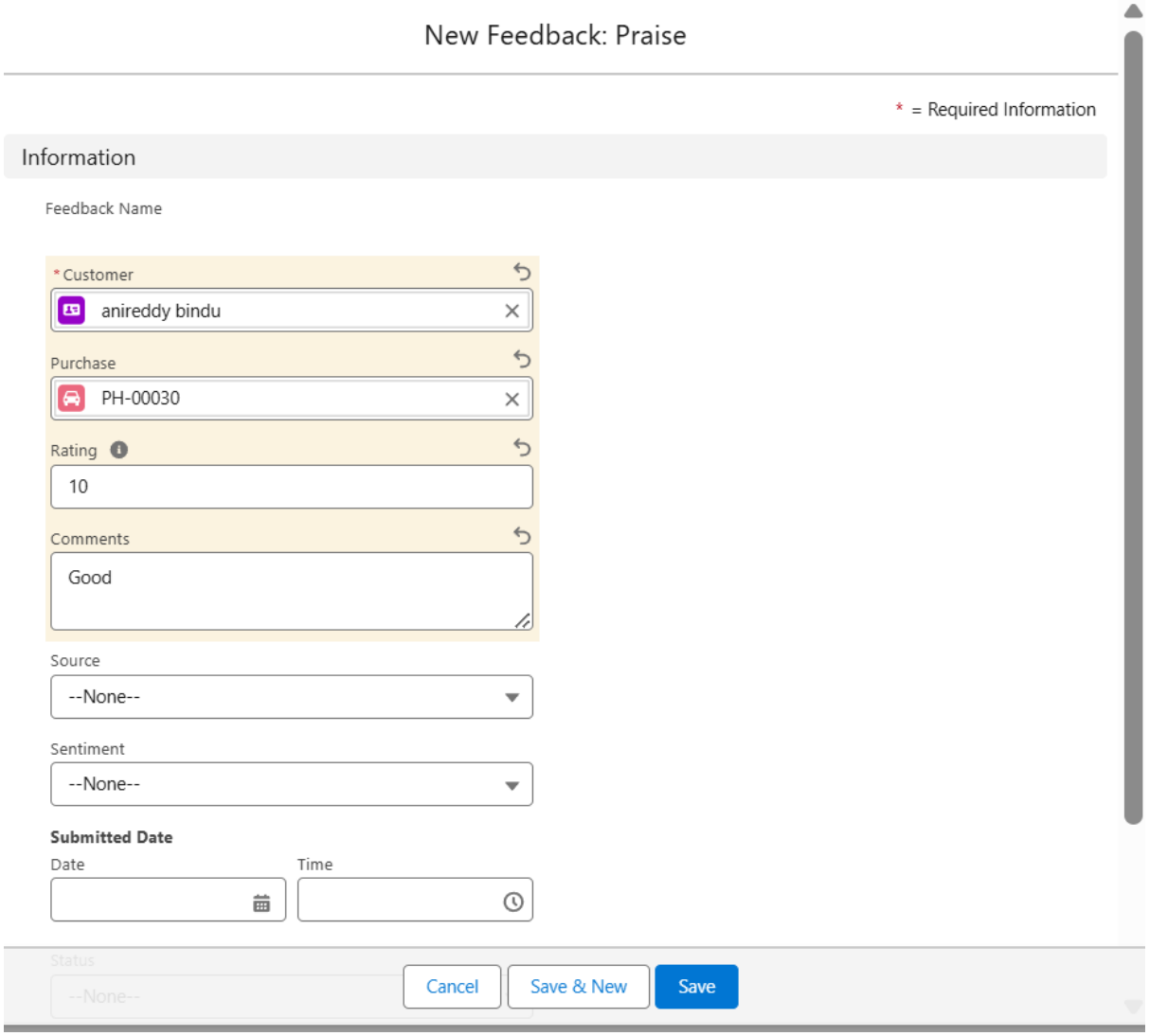
Customer Account

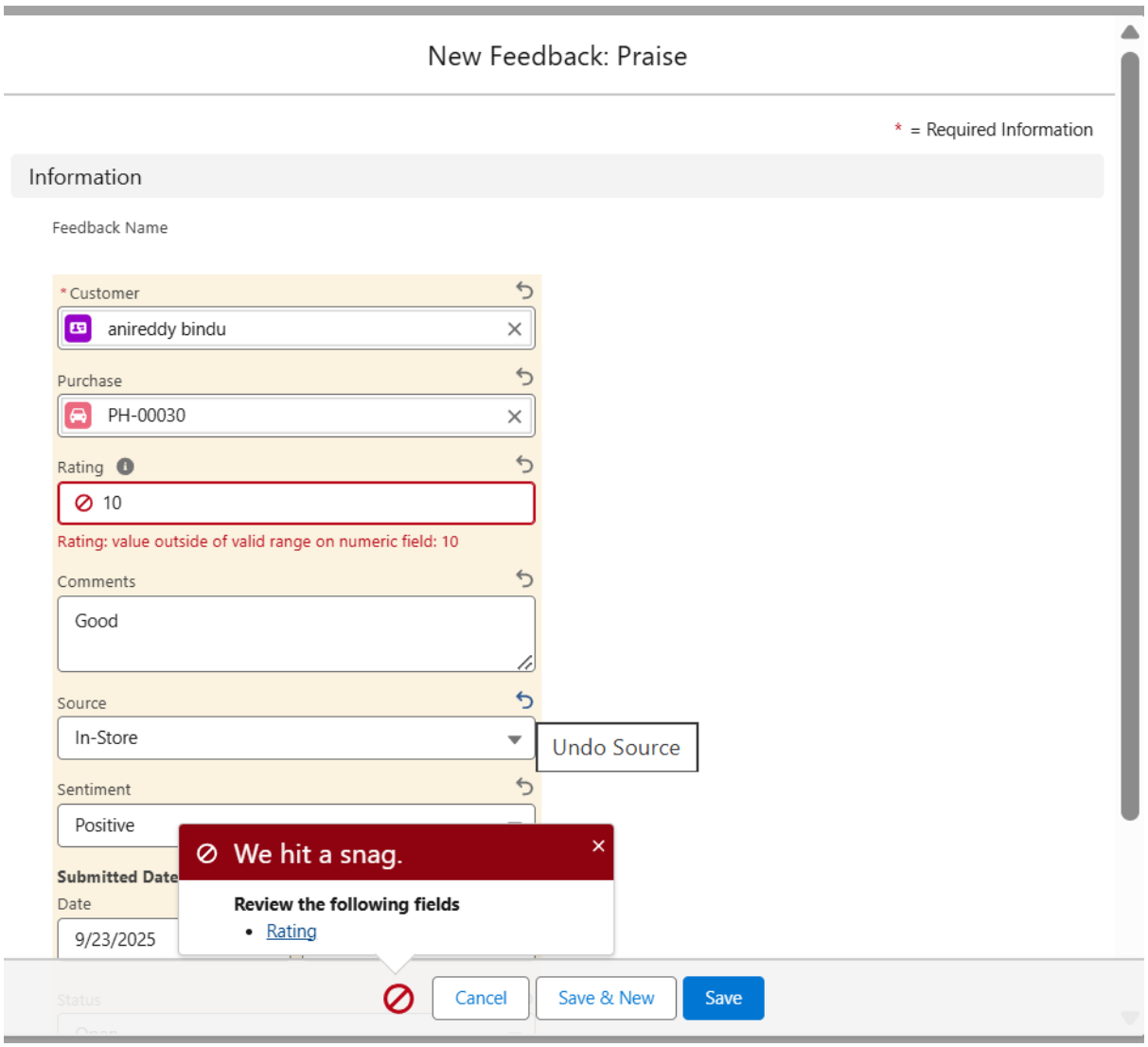
Loyalty Points
98.00

Tier
Bronze

Test Case: Feedback Submission with Validation Rule

- **Use Case / Scenario**: Feedback Rating validation.
- **Test Steps (Input)**:
    1. Go to **Feedback** → New.
    2. Enter:
        - Customer: Anireddy Bindu
        - Rating: 10
        - Comments: Service was excellent.
- **Expected Result**: Error message → "Rating must be between 1 and 5."
- **Actual Result**: Validation error appeared correctly.





Test Case: Tier Update Based on Purchase History

Use Case / Scenario: When a new Purchase History record is added, the system calculates loyalty points and automatically updates the Customer/Contact's Tier based on a Decision element in the Flow.

Test Steps (Input):

1. Ensure your Flow is active: Record-Triggered Flow on Purchase History object.
2. Confirm your Flow logic:
    - Assignment element: Calculate loyalty points = Purchase_Amount__c / 100 * 10
    - Decision element**:** Map points to tier
        - e.g.,
            - 0–499 points → Silver
            - 500–999 points → Gold
            - 1000+ points → Platinum
    - Update Records element**:** Update Customer Tier

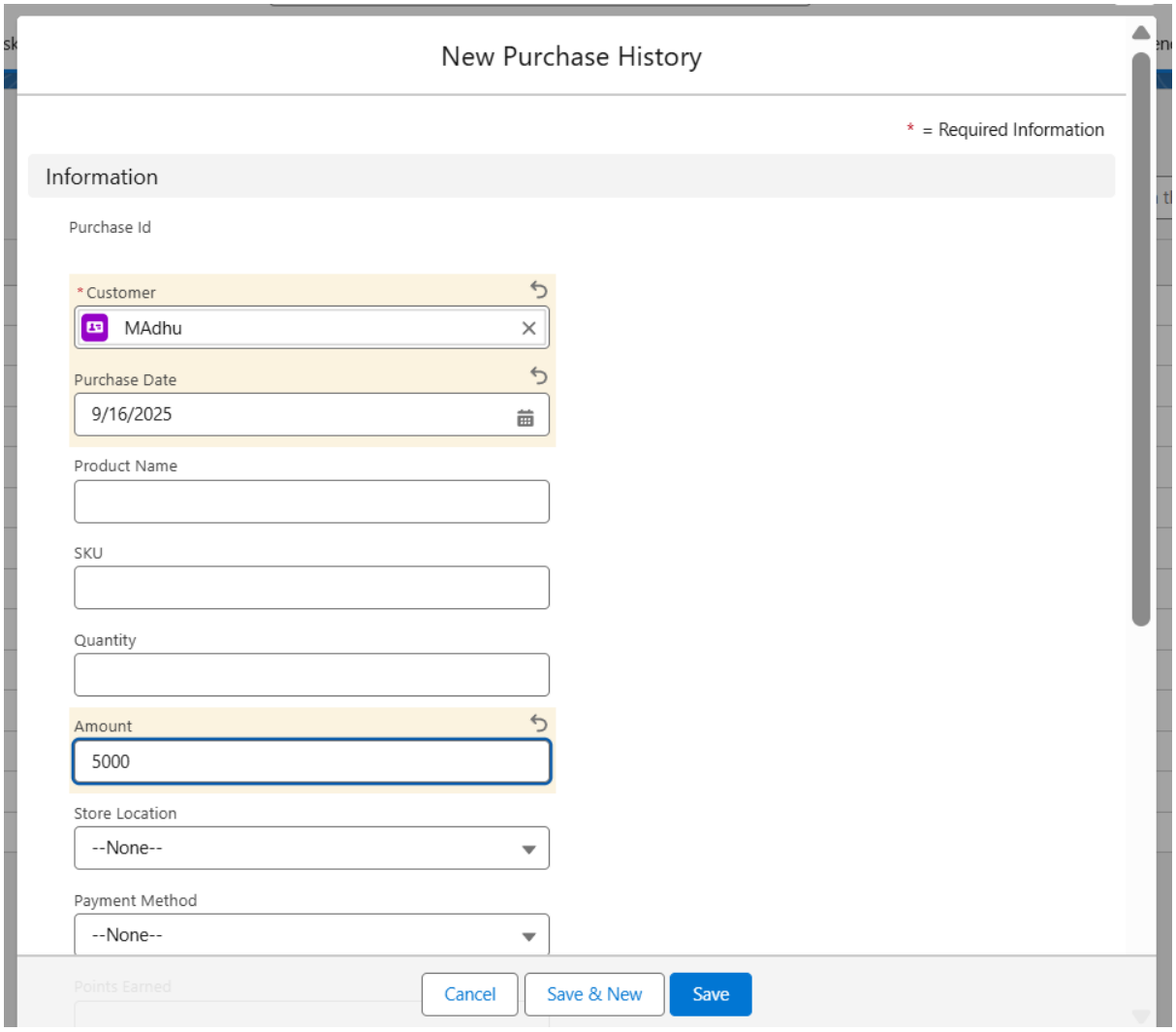
3. Create a new Purchase History record:
    - Customer: Madhu
    - Purchase Amount: ₹5,000
4. Save the record.

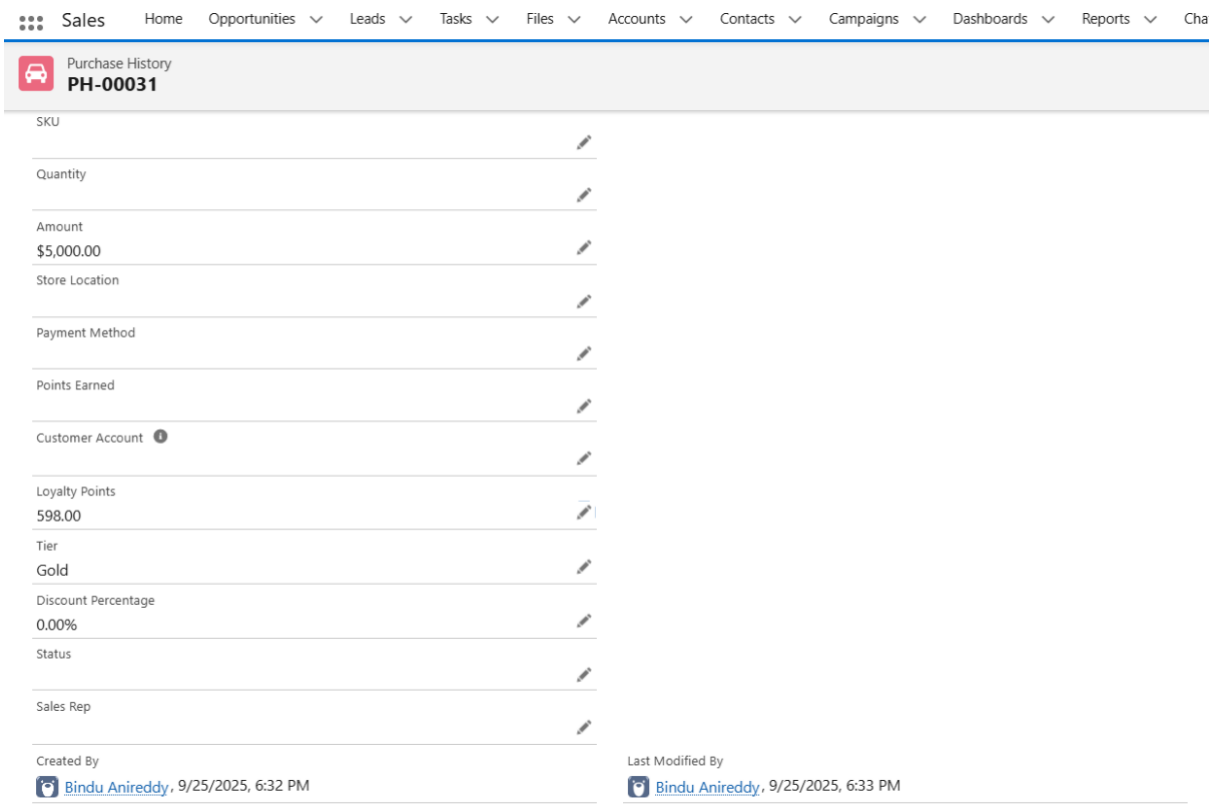
Expected Result:

- Loyalty points for this purchase = (5000 / 100) * 10 = 500 points
- Decision element evaluates 500 points → Gold Tier
- Customer record is updated: Tier = Gold
- Total loyalty points (if accumulated from previous purchases) are updated correctly

Actual Result:

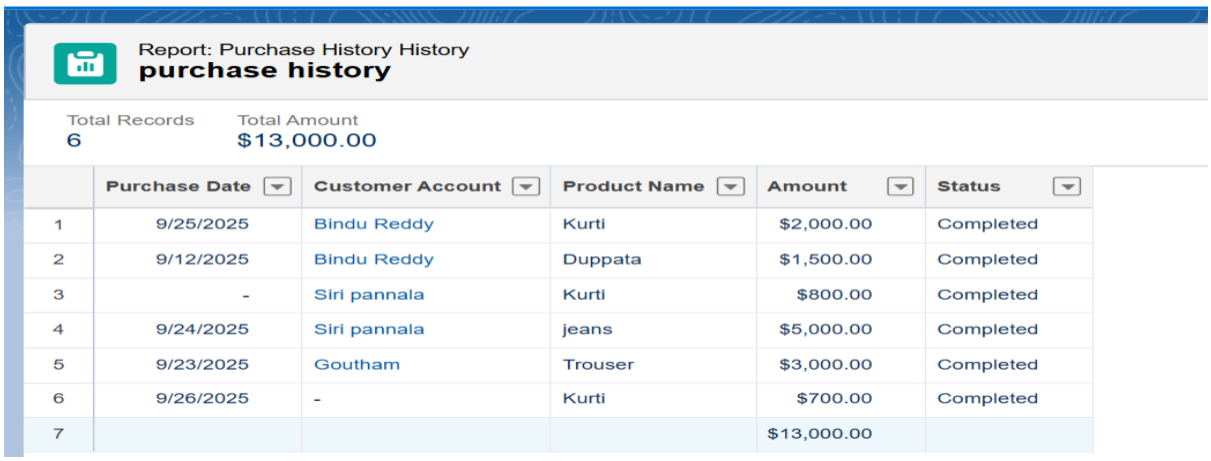
- Capture the Customer record after the purchase:
    - Total Points = 500 (or cumulative total)
    - Tier = Gold





Test Case: Report

- **Use Case / Scenario**: Calculating Total Purchases.
- **Test Steps (Input)**:
    1. Open **Reports tab** → Run Purchase History Report.
- **Expected Result**: Report shows Total Purchases.
    1. **Purchase 1**: $2000
    2. **Purchase 2**: $3000
    3. **Purchase 3**: $1500
    4. **Purchase 4**: $800
    5. **Purchase 5**: $5000
    6. **Purchase 6**: $700
    7. **Total** = $13000
- **Actual Result**: Report displays correctly.



Report: Purchase History History
purchase history

Total Records: 6
Total Amount: $13,000.00

	Purchase Date	Customer Account	Product Name	Amount	Status
1	9/25/2025	Bindu Reddy	Kurti	$2,000.00	Completed
2	9/12/2025	Bindu Reddy	Duppata	$1,500.00	Completed
3	-	Siri pannala	Kurti	$800.00	Completed
4	9/24/2025	Siri pannala	jeans	$5,000.00	Completed
5	9/23/2025	Goutham	Trouser	$3,000.00	Completed
6	9/26/2025	-	Kurti	$700.00	Completed
7				$13,000.00	

Test Case: Dashboard Filter Verification (Discount > 10)
Use Case / Scenario:
Verify that the dashboard correctly displays records with Discount > 10% and aggregates totals correctly.

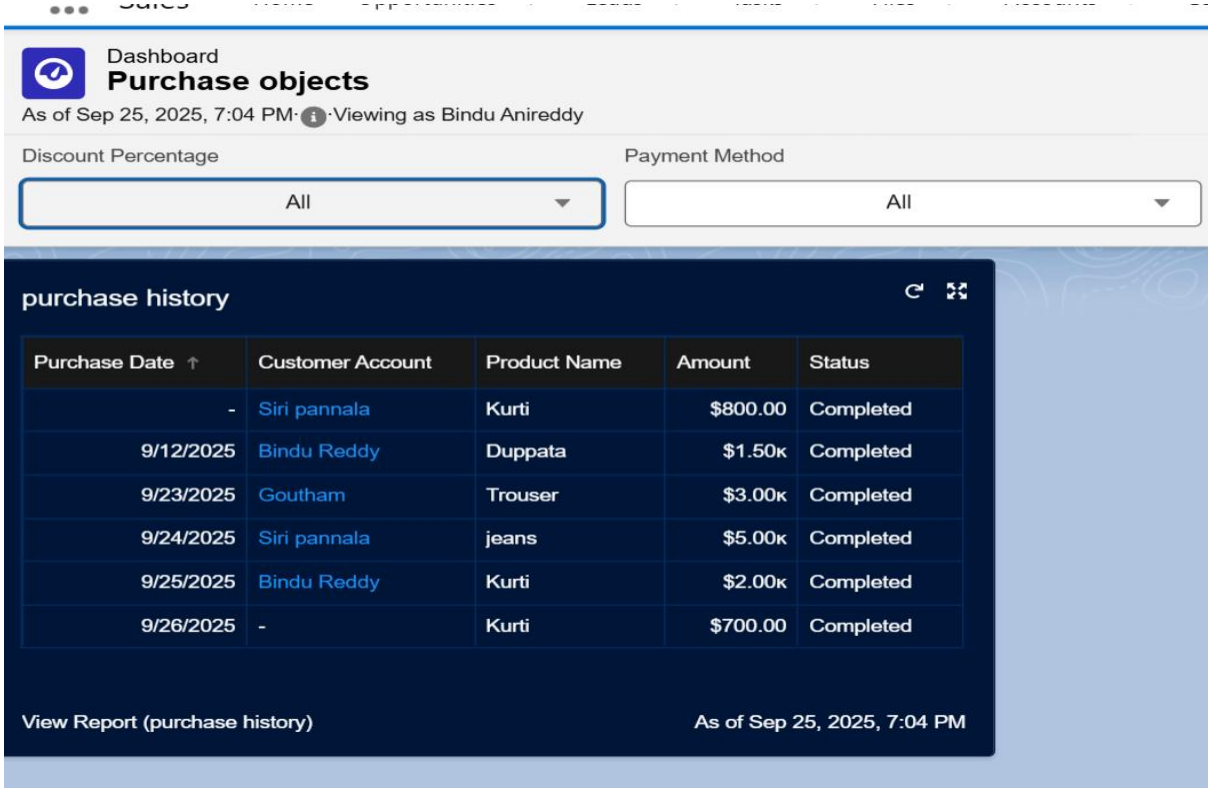
Test Steps (Input):

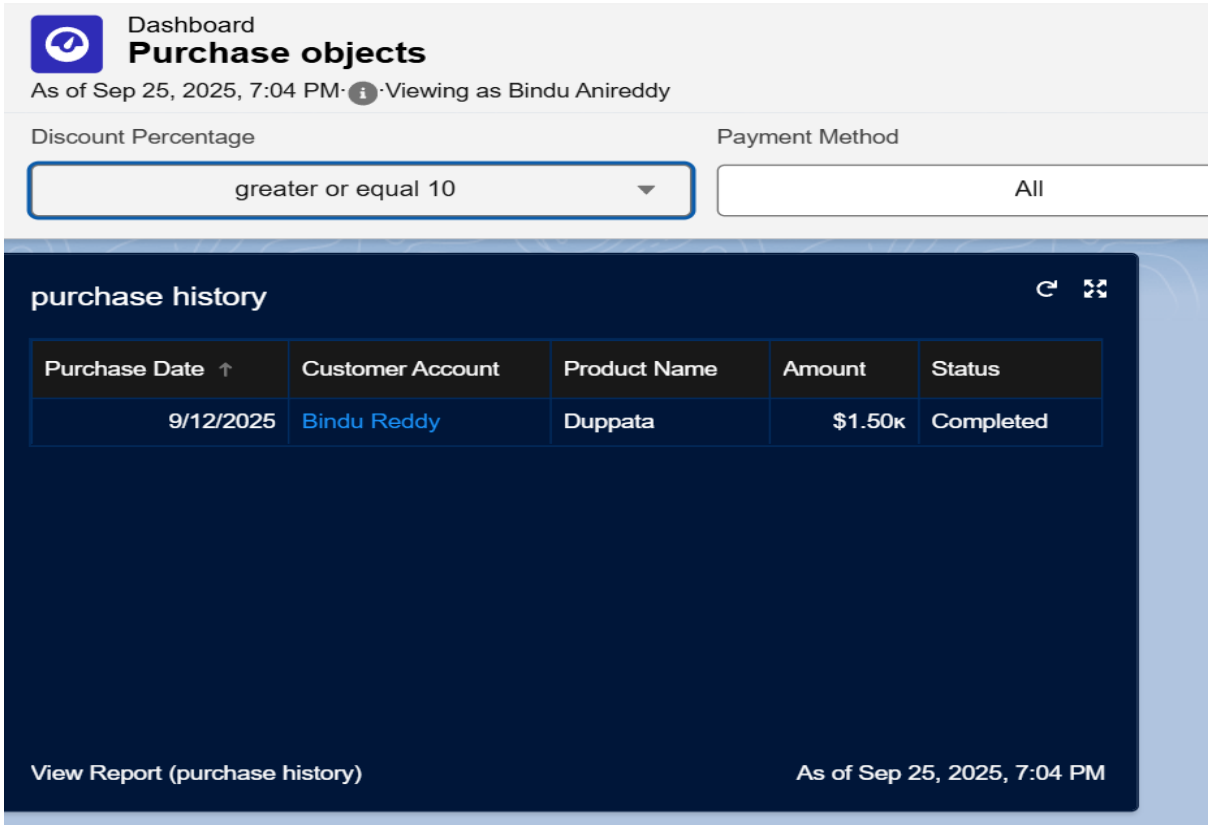
1. Navigate to your Dashboard in Salesforce.
2. Open the relevant Dashboard Component / Chart / Table.
3. Apply Dashboard Filter:
    - Field: Discount__c
    - Condition: Greater than 10
4. Ensure the data source (report) has Purchase History records with various discount values:
    - Purchase 1: Discount = 5 → should not appear
    - Purchase 2: Discount = 12 → should appear
    - Purchase 3: Discount = 20 → should appear
5. Refresh the dashboard.

Expected Result:

- Only records with Discount > 10 are displayed.
- Dashboard aggregates (sum of Purchase Amount, Total Discounts, Loyalty Points) only consider the filtered records.
- Graphs / charts reflect only the filtered data.

Actual Results:





Conclusion

The Retail Loyalty & Feedback CRM project successfully provides an integrated approach for managing customers, purchases, loyalty points, tiers, and feedback under a retail environment. The system simpifies key processes using Salesforce properties such as Flows, Apex triggers, validation rules, approval processes, and Lightning Web Components, validating accurate data management and dynamic updates. Customers' loyalty points and tiers are calculated evaluates, feedback is validated and practical tasks are generated, and approvals for high-value transactions are applied to maintain business control. Dashboards and reports provide relevant insights into purchases, loyalty metrics, and customer behavior, permitting managers to make informed decisions. Overall, the project complies its objectives by advancing operational efficiency, enhancing customer engagement, and providing a scalable platform for future growth.