

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.

The screenshot shows the 'Company Information' page in the Salesforce setup interface. The page title is 'Company Information' under the 'SETUP' tab. The main section is titled 'Retail Loyalty CRM Project'. It displays the organization's profile with various settings and metrics. Key details include:

Organization Detail		Edit	
Organization Name	Retail Loyalty CRM Project	Phone	
Primary Contact	OrgFarm EPIC	Fax	
Division		Default Locale	English (United States)
Address	United States	Default Language	English
Fiscal Year Starts In	January	Default Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)
Activate Multiple Currencies	<input type="checkbox"/>	Currency Locale	English (United States) - USD
Enable Data Translation	<input type="checkbox"/>	Used Data Space	356 KB (7%) [View]
Newsletter	<input checked="" type="checkbox"/>	Used File Space	17 KB (0%) [View]
Admin Newsletter	<input checked="" type="checkbox"/>	API Requests, Last 24 Hours	0 (15,000 max)
Hide Notices About System	<input type="checkbox"/>	Streaming API Events, Last	0 (10.000 max)

Below the table, there are sections for 'User Licenses [10+]', 'Permission Set Licenses [10+]', 'Feature Licenses [11]', and 'Usage-based Entitlements [10+]'. A 'Help for this Page' link is also present.

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 align 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.

The screenshot shows the Salesforce Setup interface with the following details:

- Header:** SETUP Business Hours
- Section Title:** Organization Business Hours
- Help:** Help for this Page ?
- Description:** Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate.
- Note:** If you enter blank business hours for a day, that means your organization does not operate on that day.
- Link:** Holidays [5]
- Table:** Business Hours Detail
- Columns:** Business Hours Name, Retail Business Hours, Time Zone, Default Business Hours (checkbox).
- Data:** A single row for "Business Hours" with the following details:

Business Hours Name	Retail Business Hours	Time Zone	Default Business Hours
Business Hours	Sunday 11:00 AM to 10:00 PM Monday 10:00 AM to 10:00 PM Tuesday 10:00 AM to 10:00 PM Wednesday 10:00 AM to 10:00 PM Thursday 10:00 AM to 10:00 PM Friday 10:00 AM to 10:00 PM Saturday 10:00 AM to 10:00 PM	(GMT+05:30) India Standard Time (Asia/Kolkata)	<input type="checkbox"/>
- Status:** Active ✓

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 aligned with corporate accounting periods.

Fiscal Year Information

Your organization can change the fiscal year start month, and specify whether the fiscal year name is set to the starting or ending year. For example, if your fiscal year starts in April 2025 and ends in March 2026, your Fiscal Year setting can be either 2025 or 2026.

Change Fiscal Year Period

Name: Retail Loyalty CRM Project
Fiscal Year Start Month: January
Fiscal Year is Based On: The ending month

5. User Setup & Licenses

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

Use Case:

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

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/>	Agent_Service	svcagent	bindusvc@domain.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Standard Platform User
<input type="checkbox"/>	Anireddy.Bindu	ani	anireddybindureddy217@agentforce.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/>	Chatter Expert	Chatter	chatty.00dg 00000bmewzuan.sgxh26xjdomz@chatter.salesforce.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Chatter Free User
<input type="checkbox"/>	EPIC_OrgFarm	OEPIC	epic.4b124a3e674c@orgfarm.salesforce.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/>	Manager_Store	storemgr	bindustoremgr@example.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Standard User
<input type="checkbox"/>	User_Integration	integ	integration@00dg 00000bmewzuan.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Analytics Cloud Integration User
<input type="checkbox"/>	User_Marketing	mktguser	bindumktg@domain.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Standard User
<input type="checkbox"/>	User_Security	sec	insightssecurity@00dg 00000bmewzuan.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Analytics Cloud Security User

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.

The screenshot shows the 'Session Settings' page under 'SETUP'. It includes sections for 'Session Timeout' (set to 2 hours) and 'Session Settings' (with various checkboxes like 'Lock sessions to the IP address from which they originated' and 'Force logout on session timeout' checked). A 'Help for this Page' link is in the top right.

The screenshot shows the 'Multi-Factor Authentication (MFA)' and 'General' settings pages. Under MFA, 'Require identity verification during multi-factor authentication (MFA) registration' is checked. Under General, several checkboxes are checked, including 'Require email confirmations for email address changes' and 'Require security tokens for API logins from callouts (API version 31.0 and earlier)'. A note at the bottom states: 'Require a high assurance level of security for sensitive operations, or block users altogether. If users already have a high assurance session after logging in, they aren't prompted to verify their identity again in the same session, even if you require high assurance for these operations. If you want to see the session levels that users are granted at login, see Session Security Levels in Session Settings.'

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.

Enable Dev Hub2 to:

- 1. Create and manage scratch orgs from the command line**
Scratch orgs are disposable Salesforce orgs that are used to support development and testing. They are fully configurable, allowing developers to emulate different Salesforce editions with different features and preferences.
- 2. View information about your scratch orgs**
Information includes details about requested scratch orgs, including whether they are active, expired or deleted.
- 3. Link namespace orgs**
You can link namespace orgs to the Dev Hub for using scratch orgs with namespaces.

Note: You can't disable this setting

Enable Dev Hub Disabled

Dev Hub

Development

Did you find what you're looking for?
Try using Global Search.

Bindu Anireddy
orgfarm-3b8243e595-dev-ed.develop.my.salesforce.com
Settings Log Out

Quick Find

USERNAMES

- datacat@gnitc.com
gnitc-14a-dev-ed.develop.my.salesforce.com
- anireddybindureddy@resourceful-fox-kqj0gg.com
resourceful-fox-kqj0gg-dev-ed.trailblaze.my.salesforce.com
- approval@gnitc.com
gnitc-b7-dev-ed.develop.my.salesforce.com

25 More Usernames

DISPLAY DENSITY

✓ Comfy
Compact

OPTIONS

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.

The screenshots show the Salesforce Object Manager interface. The top screenshot displays two custom objects: 'Contact Offer' (API Name: Contact_Offer_c, Type: Custom Object, Last Modified: 9/22/2025) and 'Offer' (API Name: Offer_c, Type: Custom Object, Last Modified: 9/21/2025). The bottom screenshot displays one custom object: 'Feedback' (API Name: Feedback_c, Type: Custom Object, Last Modified: 9/19/2025).

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Contact Offer	Contact_Offer_c	Custom Object		9/22/2025	✓
Offer	Offer_c	Custom Object		9/21/2025	▼

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Feedback	Feedback_c	Custom Object		9/19/2025	▼

Label	API Name	Type	Description	Last Modified	Deployed
Purchase History	Purchase_History__c	Custom Object		9/19/2025	<input type="button" value="▼"/>

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.

The screenshot shows the Salesforce Object Manager with the 'Feedback' object selected. The left sidebar lists various setup options: Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, and Related Lookup Filters. The main content area displays the 'Fields & Relationships' section, which lists 13 items sorted by Field Label. The table columns are: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The data includes:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Comments	Comments__c	Long Text Area(32000)		
Created By	CreatedById	Lookup(User)		
Customer	Customer__c	Master-Detail(Contact)	✓	
Feedback Name	Name	Auto Number	✓	
Last Modified By	LastModifiedById	Lookup(User)		
Purchase	Purchase__c	Lookup(Purchase History)	✓	

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.

The screenshot shows two separate object setup pages:

- Feedback Record Types:** Contains three record types: Complaint, Praise, and Suggestion. Each has a description, an active status (checked), and was modified by Bindu Anireddy on 9/22/2025 at 8:41 AM.
- Offer Record Types:** Contains three record types: Cashback, Discount, and Freebie. Each has a description, an active status (checked), and was modified by Bindu Anireddy on 9/22/2025 at various times between 12:14 AM and 12:16 AM.

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.

The screenshot shows the Page Layouts section for the Feedback object, listing four layouts:

Page Layout Name	Created By	Modified By
Complaint Layout	Bindu Anireddy, 9/22/2025, 12:04 AM	Bindu Anireddy, 9/24/2025, 5:37 AM
Feedback Layout	Bindu Anireddy, 9/19/2025, 3:13 AM	Bindu Anireddy, 9/24/2025, 5:37 AM
Praise Layout	Bindu Anireddy, 9/22/2025, 12:05 AM	Bindu Anireddy, 9/24/2025, 5:37 AM
Suggestion Layout	Bindu Anireddy, 9/22/2025, 12:04 AM	Bindu Anireddy, 9/24/2025, 5:37 AM

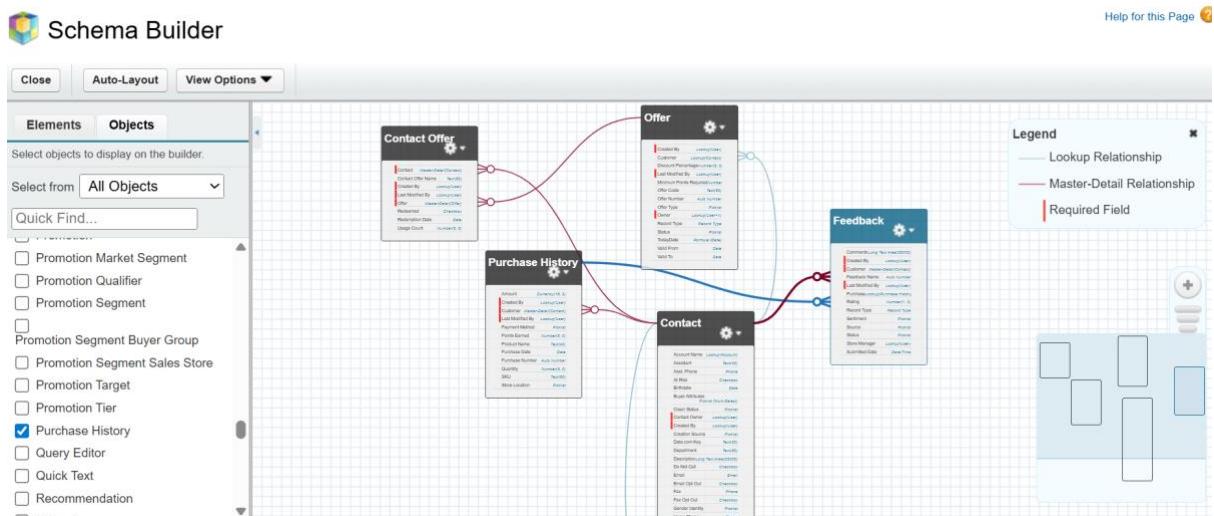
5. Compact Layouts

- 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.

The screenshot shows the Salesforce Setup interface under the Object Manager. On the left, a sidebar lists various layout types: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions. The 'Compact Layouts' option is selected and highlighted in blue. The main content area is titled 'Compact Layouts' and shows a table with two items. The first item is 'Feedback Compact Layout' with API name 'Feedback_Compact_Layout', modified by 'Bindu Anireddy' on 9/22/2025, 3:15 AM. The second item is 'System Default'. A search bar 'Quick Find' and buttons 'New' and 'Compact Layout Assignment' are at the top right of the table.

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.

The image shows two screenshots of the Salesforce Custom Field Definition Detail page. Both screenshots have a header with 'Feedback Custom Field' and the object name (Customer or Purchase). They include a 'Help for this Page' link and a 'Validation Rules [1]' link.

Customer Object Screenshot:

- Custom Field Definition Detail:** Buttons: 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

Purchase Object Screenshot:

- Custom Field Definition Detail:** Buttons: 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.

The screenshot shows the 'Contact Offer' object in the Object Manager. On the left, there's a sidebar with links like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, and Record Types. The main area is titled 'Fields & Relationships' with a sub-header '8 Items, Sorted by Field Label'. It contains a table with columns: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The table rows are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Contact	Contact_c	Master-Detail(Contact)		✓
Contact Offer Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Offer	Offer__c	Master-Detail(Offer)		✓
Redeemed	Redeemed__c	Checkbox		

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.

The screenshot shows the 'External Objects' section in the Setup menu. At the top, there's a 'New External Object' button. Below it is a table with columns: Action, Label, Deployed, External Data Source, and Description. The table rows are:

Action	Label	Deployed	External Data Source	Description
Edit Erase	Category	<input type="checkbox"/>	SupplierCatalog	Categories
Edit Erase	Product	<input type="checkbox"/>	SupplierCatalog	Products

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 than 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.

Purchase History Validation Rule Help for this Page 

[Back to Purchase History](#)

Validation Rule Detail		Edit	Clone
Rule Name	Prevent_Negative_Amount	Active	<input checked="" type="checkbox"/>
Error Condition Formula	Amount_c < 0		
Error Message	Purchase Amount cannot be negative.	Error Location	Amount
Description			
Created By	Bindu Anireddy, 9/24/2025, 1:19 PM	Modified By	Bindu Anireddy, 9/24/2025, 1:19 PM
Edit Clone			

Offer Validation Rule Help for this Page 

[Back to Offer](#)

Validation Rule Detail		Edit	Clone
Rule Name	Discount_Range_Check	Active	<input checked="" type="checkbox"/>
Error Condition Formula	OR(Discount_Percentage_c < 0, Discount_Percentage_c > 100)		
Error Message	Discount must be between 0 and 100	Error Location	Discount Percentage
Description			
Created By	Bindu Anireddy, 9/21/2025, 9:24 PM	Modified By	Bindu Anireddy, 9/21/2025, 9:24 PM
Edit Clone			

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.

Workflow Rule Detail

Rule Name	High_Value_Purchase_Notification	Object	Purchase History	
Active	<input checked="" type="checkbox"/>	Evaluation Criteria	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria	
Description				
Rule Criteria	Purchase History: Amount GREATER THAN 10000			
Created By	Bindu Anireddy, 9/25/2025, 12:12 PM		Modified By	Bindu Anireddy, 9/25/2025, 12:15 PM

Workflow Actions

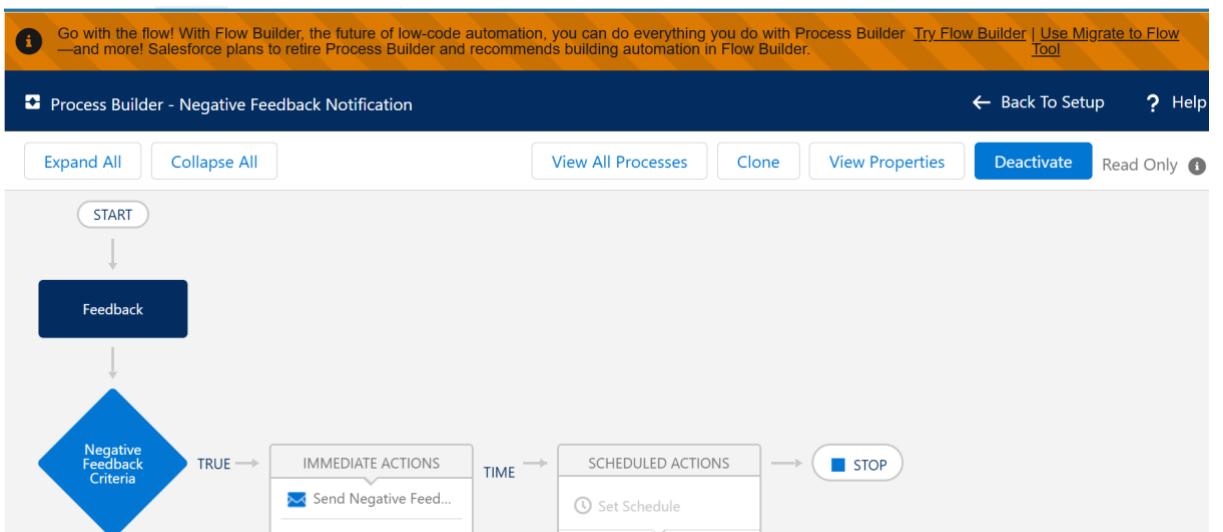
Immediate Workflow Actions

Type	Description
Email Alert	High points

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:

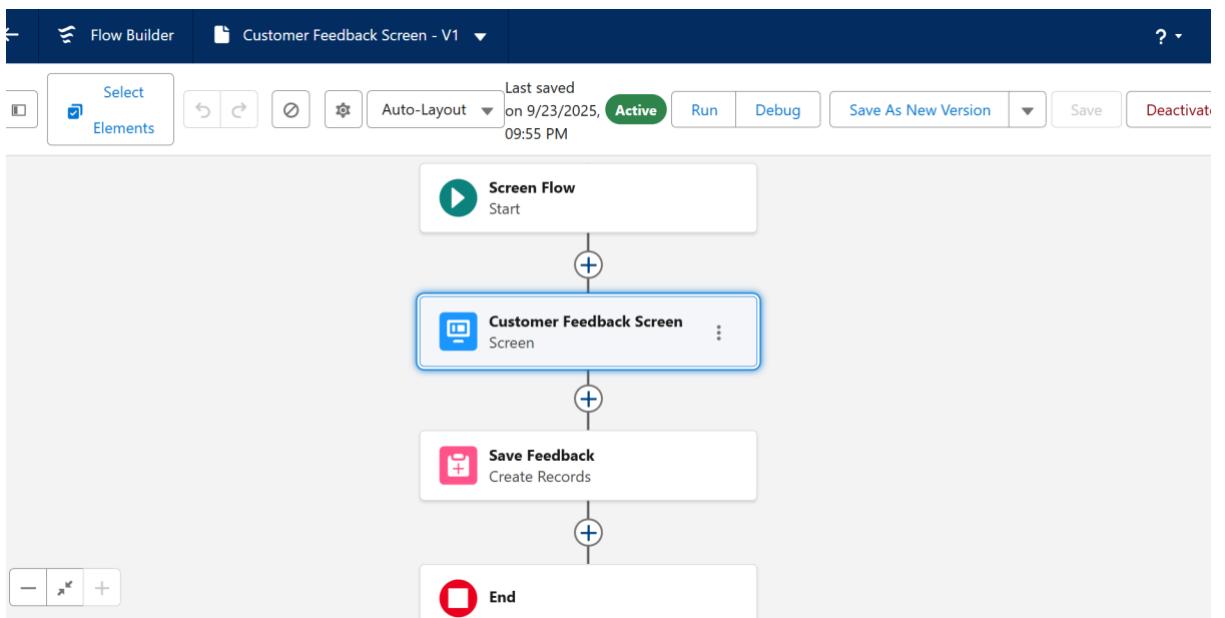
When a customer rating is **less than 3**, the record can be sent for manager sanction. This ensures poor experiences are examined and addressed rapidly to maintain service quality.

The screenshot shows the Salesforce Approval Processes interface. At the top, there's a blue header bar with a gear icon and the word "SETUP". Below it, the main title is "Approval Processes" and the specific process is "Feedback: Feedback Approval". There's a "Help for this Page" link with a question mark icon. A backlink "« Back to Approval Process List" is also present. The main content area is titled "Process Definition Detail" and shows the following details:

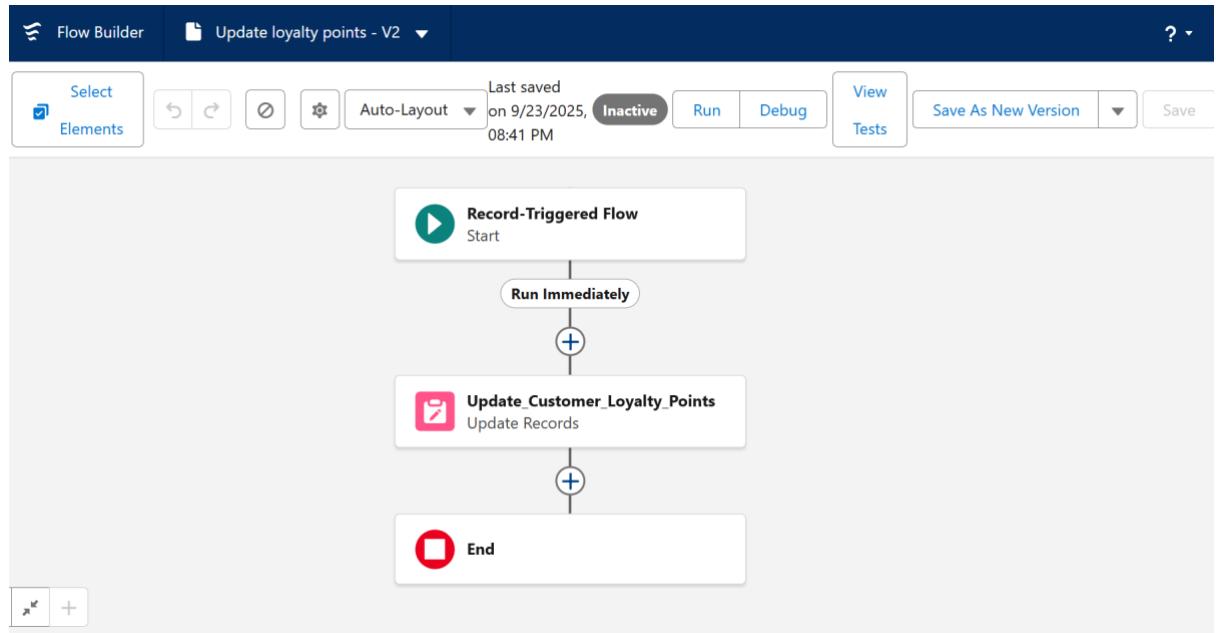
Process Name	Feedback Approval	Active
Unique Name	Feedback_Approval	Next Automated Approver Determined By
Description	Manager of Record Submitter	
Entry Criteria	Feedback: Rating LESS OR EQUAL 2	
Record Editability	Administrator ONLY	Allow Submitters to Recall Approval Requests
Approval Assignment Email Template	<u>Negative Feedback Notification</u>	

5. Flow Builder

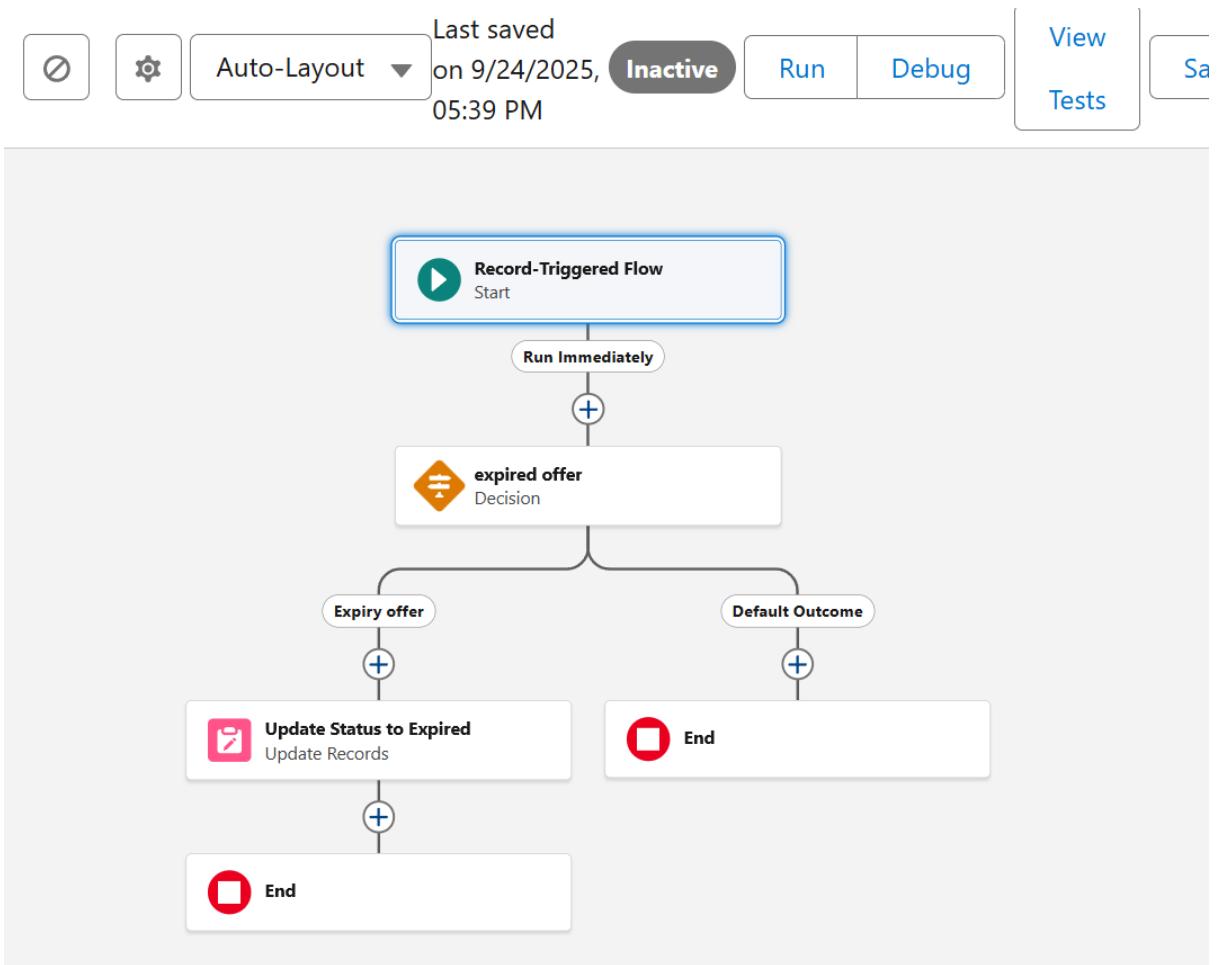
- **Screen Flow:** Design a guided flow for agents to gather customer feedback (Rating, Comments) directly while service calls.



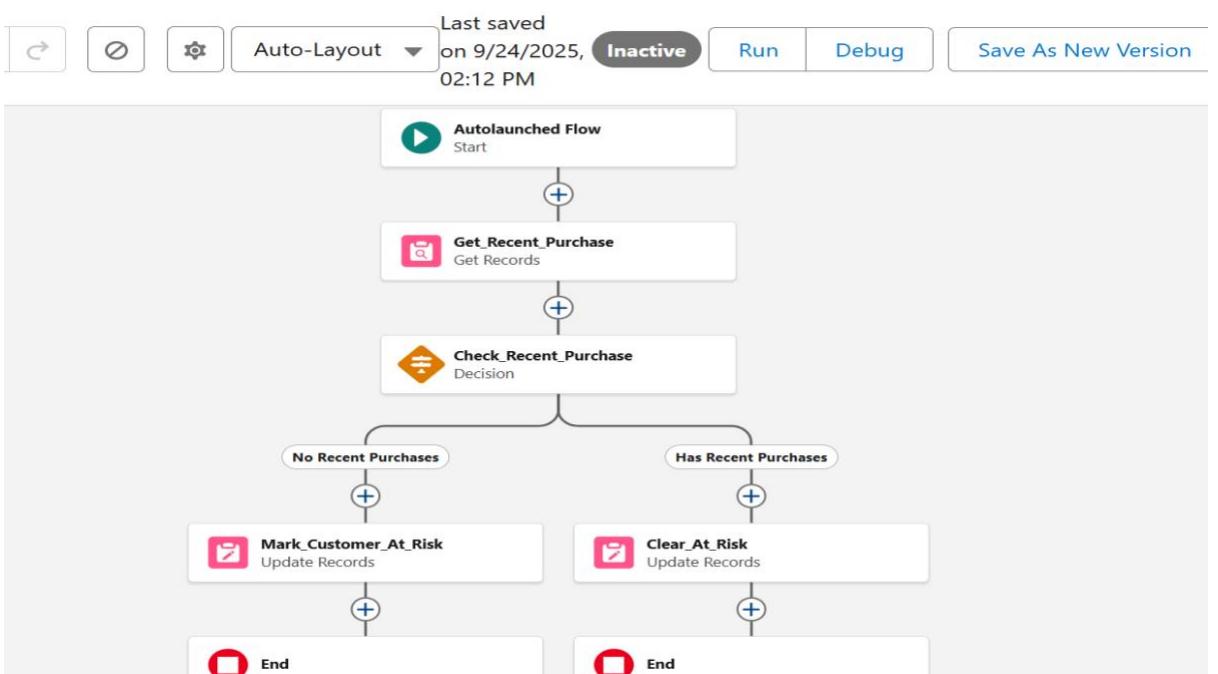
- **Record-Triggered Flow:** When a new purchase is recorded, automatically allot loyalty points to the customer.



- **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 forwarded to the support manager. This ensures instant action is taken to address the problem and retain customer trust.

The screenshot shows the Salesforce Setup interface with the 'Email Alerts' section selected. The page title is 'Email Alert Negative_Feedback_Alert'. It displays the 'Email Alert Detail' for this specific alert. The alert's description is 'Negative_Feedback_Alert', it uses the 'Negative Feedback Notification' template, and its object is 'Feedback'. The recipient is set to 'User: Store Manager'. The alert was created by Bindu Anireddy on 9/24/2025 at 3:49 AM and modified by the same user on 9/24/2025 at 4:36 AM. There are standard edit, delete, and clone buttons at the bottom of the detail section.

This screenshot shows another instance of the Email Alert detail page. The alert is named 'High points' and uses the 'Negative Feedback Notification' template. Its object is 'Purchase History'. The recipient is 'User: Store Manager'. It was created by Bindu Anireddy on 9/25/2025 at 12:16 PM and modified by the same user on the same date and time. The layout is identical to the first screenshot, with fields for description, unique name, from email address, recipients, additional emails, and creation/modification details, along with edit, delete, and clone buttons.

7. Field Updates

Use Case:

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

The screenshot shows the 'Field Update Detail' page for a field named 'Loyalty'. The page includes a header with a gear icon and the word 'SETUP', followed by 'Field Updates'. Below the header, the section title is 'Field Update Loyalty'. A sub-section title 'Field Update Detail' is shown above a table of field details. The table has two columns: 'Name' and 'Value'. The rows are:

Name	Loyalty
Unique Name	Loyalty
Description	
Object	Purchase History
Field to Update	Purchase History: Tier
Field Data Type	Picklist
Re-evaluate Workflow Rules after Field Change	<input type="checkbox"/>
New Field Value	Gold

At the bottom of the table are 'Edit' and 'Delete' buttons. At the very bottom of the page are 'Edit', 'Delete', and 'Clone' buttons.

8. Tasks

Use Case:

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

The screenshot shows the 'Workflow Task Detail' page for a task titled 'Follow up on Pending Payment'. The page includes a header with a gear icon and the word 'SETUP', followed by 'Tasks'. Below the header, the section title is 'Task Follow up on Pending Payment'. A sub-section title 'Workflow Task Detail' is shown above a table of task details. The table has two columns: 'Object' and 'Value'. The rows are:

Object	Purchase History
Assigned To	User : Service Agent
Subject	Follow up on Pending Payment
Unique Name	Follow_up_on_Pending_Payment
Due Date	Purchase History: Created Date + 3 days
Comments	
Created By	Bindu Anireddy, 9/25/2025, 12:55 PM
Modified By	Bindu Anireddy, 9/25/2025, 12:55 PM

At the bottom of the table are 'Edit', 'Delete', and 'Clone' buttons. At the very bottom of the page are 'Edit', 'Delete', and 'Clone' buttons.

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.

NOTIFICATION NAME	API NAME	NAMESPACE	DESKTOP	MOBILE
enablement_coaching_feedback_ready	enablement_coaching_feedback_ready		✓	▼
Feedback Escalation Alert	Feedback_Escalation_Alert		✓	✓ ▼

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.

Name	LoyaltyPointsHandler	Status	Active
Namespace Prefix		Code Coverage	0% (0/9)
Created By	Bindu Anireddy , 9/25/2025, 1:04 PM	Last Modified By	Bindu Anireddy , 9/25/2025, 1:04 PM

```

1 public class LoyaltyPointsHandler {
2
3     // Method to add loyalty points based on purchase amount
4     public static void addPoints(List<Purchase_History__c> purchases) {
5         List<Contact> contactsToUpdate = new List<Contact>();
6
7         for(Purchase_History__c ph : purchases) {
8             if(ph.Amount__c > 0 && ph.Customer__c != null) {
9                 // Retrieve customer
10                Contact c = [SELECT Id, Loyalty_Points__c FROM Contact WHERE Id = :ph.Customer__c LIMIT 1];

```

```

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 }

```

[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.

The screenshot shows the Apex Trigger Detail page for a trigger named "PurchaseHistoryTrigger". The page includes a header with a gear icon labeled "SETUP" and "Apex Triggers", and a sub-header with "Apex Trigger" and "PurchaseHistoryTrigger". It features a "Help for this Page" link with a question mark icon. Below the header is a table with columns for Name, sObject Type, Code Coverage, Status, Created By, Last Modified By, and Namespace Prefix. The "Name" column shows "PurchaseHistoryTrigger", "sObject Type" shows "Purchase History", "Code Coverage" shows "0% (0/2)", "Status" shows "Active", "Created By" shows "Bindu Anireddy, 9/25/2025, 1:10 PM", "Last Modified By" shows "Bindu Anireddy, 9/25/2025, 1:10 PM", and "Namespace Prefix" is empty. At the bottom, there are tabs for "Apex Trigger", "Version Settings", and "Trace Flags", with "Apex Trigger" being the active tab. The code pane displays the trigger definition:

```

1trigger PurchaseHistoryTrigger on Purchase_History__c (after insert) {
2    if(Trigger.isAfter && Trigger.isInsert){
3        LoyaltyPointsHandler.addPoints(Trigger.new);
4    }
5}

```

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.

SETUP

Apex Classes

Apex Class LoyaltyPointsHandler

Help for this Page 

Apex Class Detail		Edit	Delete	Download	Security	Show Dependencies	
Name	LoyaltyPointsHandler	Status	Active				
Namespace Prefix		Code Coverage	0% (0/9)				
Created By	Bindu Anireddy , 9/25/2025, 1:04 PM	Last Modified By	Bindu Anireddy , 9/25/2025, 1:04 PM				

Class Body Class Summary Version Settings Trace Flags

```

1 public class LoyaltyPointsHandler {
2
3     // Method to add loyalty points based on purchase amount
4     public static void addPoints(List<Purchase_History__c> purchases) {
5         List<Contact> contactsToUpdate = new List<Contact>();
6
7         for(Purchase_History__c ph : purchases) {
8             if(ph.Amount__c > 0 && ph.Customer__c != null) {
9                 // Retrieve customer
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 }
```

Edit Delete Download Security Show Dependencies

SETUP

Apex Triggers

Apex Trigger PurchaseHistoryTrigger

Help for this Page 

Apex Trigger Detail		Edit	Delete	Download	Show Dependencies
Name	PurchaseHistoryTrigger	sObject Type	Purchase History		
Code Coverage	0% (0/2)	Status	Active		
Created By	Bindu Anireddy, 9/25/2025, 1:10 PM	Last Modified By	Bindu Anireddy, 9/25/2025, 1:10 PM		
Namespace Prefix					

Apex Trigger Version Settings Trace Flags

```

1 trigger PurchaseHistoryTrigger on Purchase_History__c (after insert) {
2     if(Trigger.isAfter && Trigger.isInsert){
3         LoyaltyPointsHandler.addPoints(Trigger.new);
4     }
5 }
```

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.

The screenshot shows the Salesforce Query Editor interface. At the top, there are several tabs: PurchaseHistoryTrigger.apxt, PurchaseHistoryHandler.apxc, OfferService.apxc, Contact@1:54 AM, Contact@1:55 AM, and Contact@1:56 AM (which is highlighted). Below the tabs, a SOQL query is displayed:

```
SELECT Id, Name, Email FROM Contact WHERE LastName = 'smith'
```

The results section shows a table with three rows:

Id	Name	Email
003gL00000CqHXPQA3	little Smith	
003gL00000DB1bfQAH	Smith	
003gL00000DBJmFQAX	Alice Smith	alice.smith@example.com

Below the results, there are buttons for Query Grid, Save Rows, Insert Row, Delete Row, and Refresh Grid. To the right, there are links for Access in Salesforce, Create New, Open Detail Page, and Edit Page. The bottom navigation bar includes Logs, Tests, Checkpoints, Query Editor (which is selected), View State, Progress, and Problems. A history panel on the right shows the executed query:

```
SELECT Id, Name, Email FROM Contact WHERE LastName = 'smith'
```

At the bottom, a note says "Any query errors will appear here..."

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.

```

1  public class CollectionsDemo {
2
3      public static void demoCollections() {
4          // ----- LIST -----
5          List<String> fruits = new List<String>{'Apple', 'Banana', 'Orange', 'Apple'};
6          System.debug('List: ' + fruits);
7
8          // ----- SET -----
9          Set<String> uniqueFruits = new Set<String>{'Apple', 'Banana', 'Orange', 'Apple'};
10         System.debug('Set: ' + uniqueFruits);
11
12         // ----- MAP -----
13         Map<String, Integer> fruitCount = new Map<String, Integer>();
14         for(String fruit : fruits){
15             if(fruitCount.containsKey(fruit)){
16                 fruitCount.put(fruit, fruitCount.get(fruit) + 1);
17             } else {
18                 fruitCount.put(fruit, 1);
19             }
20         }
21         System.debug('Map (fruit counts): ' + fruitCount);
22     }
23 }

```

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}

Logs						
User	Application	Operation	Time	Status	Read	Size
Bindu Anireddy	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.

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

Contact@1:55 AM [] Contact@1:56 AM [] CollectionsDemo.apxc [] Log executeAnonymous @9/26/2025, 2:04:57 AM [] RetailControlDemo.apxc [] Log executeAnonymous @9/26/2025, 2:14:59 AM []

Code Coverage: None ▾ API Version: 64 ▾

```

1 public class RetailControlDemo {
2
3     public static void demoRetailControl() {
4         // ----- IF-ELSE: Discount Logic -----
5         Decimal totalAmount = 1200;
6         if(totalAmount >= 1000) {
7             System.debug('Discount: 20%');
8         } else if(totalAmount >= 500) {
9             System.debug('Discount: 10%');
10        } else {
11            System.debug('No Discount');
12        }
13
14        // ----- FOR LOOP: Display Items in Cart -----
15        List<String> cartItems = new List<String>{'Shirt', 'Jeans', 'Shoes'};
16        for(String item : cartItems) {
17            System.debug('Cart Item: ' + item);
18        }
19
20        // ----- WHILE LOOP: Process Pending Orders -----
21        Integer pendingOrders = 3;
22        while(pendingOrders > 0) {
23            System.debug('Processing order, remaining: ' + pendingOrders);
24            pendingOrders--;
25        }
26
27        // ----- SWITCH: Order Status Handling -----
28        String orderStatus = 'Shipped';
29        switch on orderStatus {
30            when 'Pending' {
31                System.debug('Order not yet processed');
32            }
33            when 'Shipped' {
34                System.debug('Order is on the way');
35            }
36            when 'Delivered' {
37                System.debug('Order delivered to customer');
38            }
39            when else {
40                System.debug('Unknown status');
41            }
42        }
43    }
44 }
```

Contact@1:55 AM [] Contact@1:56 AM [] CollectionsDemo.apxc [] Log executeAnonymous @9/26/2025, 2:04:57 AM [] RetailControlDemo.apxc [] Log executeAnonymous @9/26/2025, 2:14:59 AM []

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

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

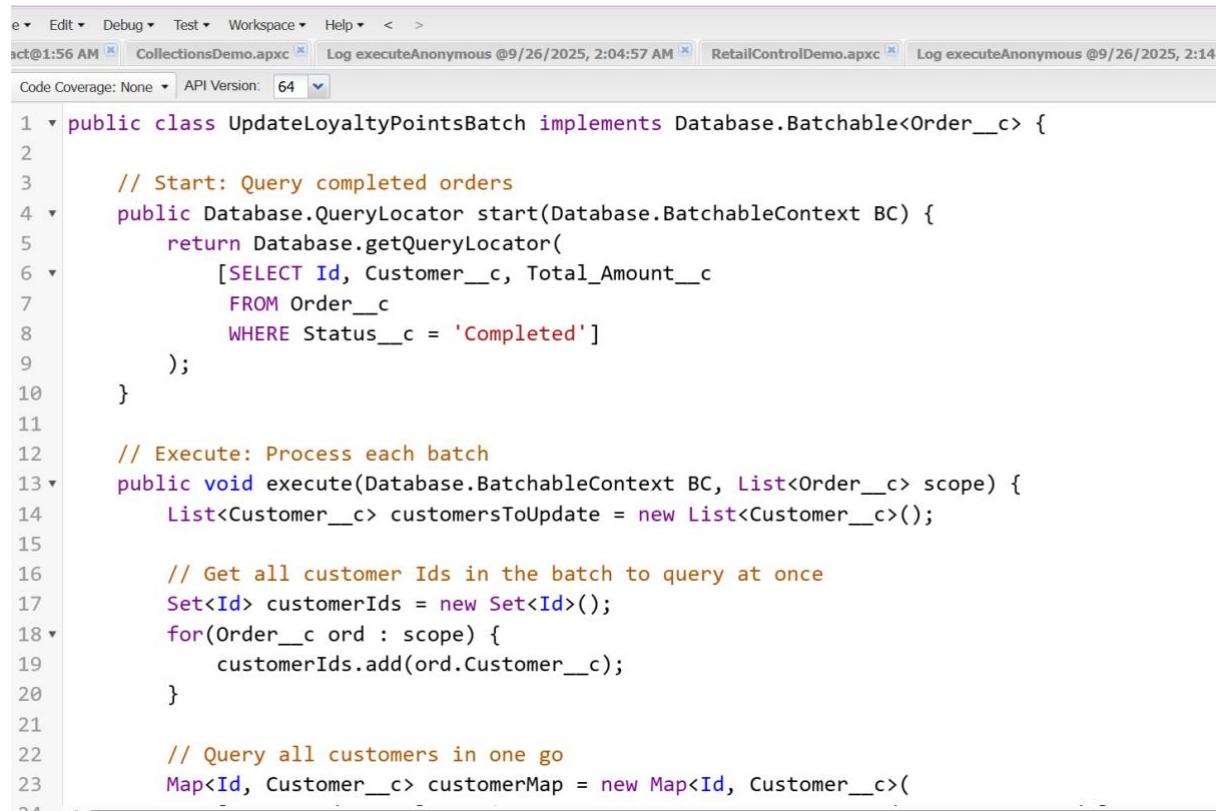
Logs

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.



The screenshot shows the Salesforce IDE interface with the code for the `UpdateLoyaltyPointsBatch` class. The code implements the `Database.Batchable<Order__c>` interface. It starts by querying completed orders from the `Order__c` object. Then, it processes each batch to update customer loyalty points. The code uses a `Set<Id>` to store customer IDs and a `Map<Id, Customer__c>` to map them to their respective records.

```
1 public class UpdateLoyaltyPointsBatch implements Database.Batchable<Order__c> {
2
3     // Start: Query completed orders
4     public Database.QueryLocator start(Database.BatchableContext BC) {
5         return Database.getQueryLocator(
6             [SELECT Id, Customer__c, Total_Amount__c
7              FROM Order__c
8              WHERE Status__c = 'Completed']
9         );
10    }
11
12    // Execute: Process each batch
13    public void execute(Database.BatchableContext BC, List<Order__c> scope) {
14        List<Customer__c> customersToUpdate = new List<Customer__c>();
15
16        // Get all customer Ids in the batch to query at once
17        Set<Id> customerIds = new Set<Id>();
18        for(Order__c ord : scope) {
19            customerIds.add(ord.Customer__c);
20        }
21
22        // Query all customers in one go
23        Map<Id, Customer__c> customerMap = new Map<Id, Customer__c>(
24
```

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.

no.apxc Log executeAnonymous @9/26/2025, 2:04:57 AM RetailControlDemo.apxc Log execu
e Coverage: None API Version: 64

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

SETUP
Apex Jobs

[Click here to go to the new batch jobs page](#)

Apex Jobs

Monitor the status of all Apex jobs, and optionally, abort jobs that are in progress.

Percent of Asynchronous Apex Used: 0%
 You have currently used 1 asynchronous Apex operations out of an allowed 24-hour organization limit of 250,000. To learn about how this limit is calculated, see the [Lightning Platform Apex Limits](#) topic.

Action	Submitted Date	Job Type	Status	Status Detail	Total Batches	Batches Processed	Failures	Submitted By	Completion Date	Apex Class
9/25/2025, 2:02 PM	Queueable	Completed			0	0	0	Anireddy, Bindu	9/25/2025, 2:02 PM	SimpleQueueabl

Scheduled Apex

Use Case:

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

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

PurchaseHistoryTrigger.apxc PurchaseHistoryHandler.apxc OfferService.apxc **RetailScheduledJob.apxc**

Code Coverage: None API Version: 64

```
1 global class RetailScheduledJob implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         // Example: Update all Orders with Status = 'Pending' to 'Processing'
4         List<Order__c> ordersToUpdate = [SELECT Id, Status__c FROM Order__c WHERE Status__c = 'Pending'];
5         for(Order__c ord : ordersToUpdate) {
6             ord.Status__c = 'Processing';
7         }
8         update ordersToUpdate;
9     }
10 }
11
```



Scheduled Jobs

Del	calculate_total_purchases-3	Anireddy, Bindu	9/24/2025, 11:35 PM	9/25/2025, 10:00 PM	Scheduled Flow	08egL00000Ccy4S
Del	CommIncrementalSitemapJob-00DgL00000BmEWz-0DMgL000000HsAU	Anireddy, Bindu	9/25/2025, 2:46 AM	9/25/2025, 7:57 PM	Sitemap SEO Incremental Job	08egL00000CdUlK
Del	CommSitemapJob-00DgL00000BmEWz-0DMgL000000HsAU	Anireddy, Bindu	9/25/2025, 2:46 AM	9/27/2025, 6:30 PM	Sitemap SEO Generation Job	08egL00000CdUlK
Del	Metalytics Data Loader Job for Org : 00DgL00000BmEWz	User, Integration	9/15/2025, 7:44 AM	9/24/2025, 9:34 PM	Autonomous Data Loader Job	08egL00000C63Nn
	Program Milestone Computation Cron Job	Process, Automated	9/15/2025, 7:44 AM	9/25/2025, 12:00 PM	Program Milestone Computation Cron Job	08egL00000C63NI
	Program Status Update Cron Job	Process, Automated	9/15/2025, 7:44 AM	9/25/2025, 5:00 AM	Program Status Update Cron Job	08egL00000C63Nm
Manage Del Pause Job	RetailOrderProcessing	Anireddy, Bindu	9/25/2025, 2:14 PM	10/2/2025, 10:00 AM	Scheduled Apex	08egL00000Ce07M

Future Methods

Use Case:

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

```

1 public class RetailFutureJob {
2
3     // @future annotation allows this method to run asynchronously
4     @future
5     public static void updateOrderStatus(List<Id> orderIds) {
6         List<Order__c> ordersToUpdate = [SELECT Id, Status__c FROM Order__c WHERE Id IN :orderIds];
7         for(Order__c ord : ordersToUpdate) {
8             ord.Status__c = 'Processed';
9         }
10        update ordersToUpdate;
11    }
12 }
```

Exception Handling

Use Case:

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

```

1  public class RetailExceptionDemo {
2    public static void processOrders() {
3      try {
4        // Fetch pending orders
5        List<Order__c> orders = [SELECT Id, Status__c FROM Order__c WHERE Status__c = 'Pending'];
6
7        // Use System.assert to check if orders exist
8        System.assert(!orders.isEmpty(), 'No pending orders found.');
9
10       // Update orders
11      for(Order__c ord : orders) {
12        ord.Status__c = 'Processing';
13      }
14      update orders;
15
16      System.debug('Orders updated successfully.');
17
18    } catch (Exception e) {
19      System.debug('Handled Exception: ' + e.getMessage());
20    } finally {
21      System.debug('Process completed (finally block executed).');
22    }
23  }
24}
25

```

Test Classes

Use Case:

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

```

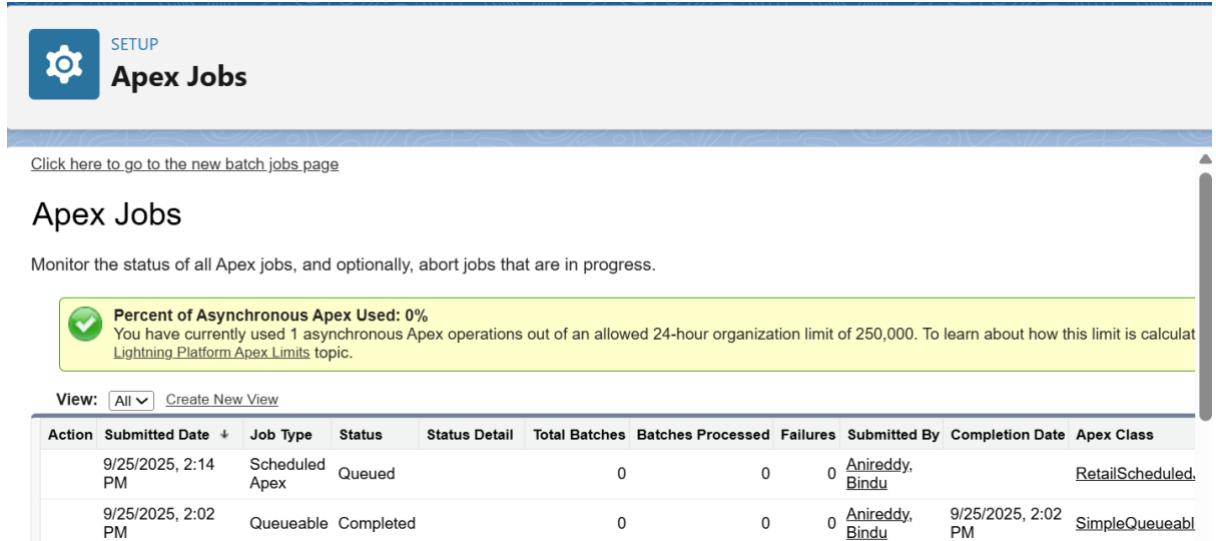
Code Coverage: None | API Version: 64
1 @isTest
2 public class RetailOrderTest {
3
4   @isTest
5   static void testProcessOrders() {
6     // Create test data
7     Order__c testOrder = new Order__c(Name='Test Order', Status__c='Pending');
8     insert testOrder;
9
10    // Call the method to test
11    RetailExceptionDemo.processOrders();
12
13    // Verify results
14    testOrder = [SELECT Status__c FROM Order__c WHERE Id = :testOrder.Id];
15    System.assertEquals('Processing', testOrder.Status__c, 'Order status should be updated to Processing');
16  }
17}
18

```

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.



The screenshot shows the 'Apex Jobs' page under the 'SETUP' tab. At the top, there's a button to 'Click here to go to the new batch jobs page'. Below it, a section titled 'Apex Jobs' displays a message: 'Percent of Asynchronous Apex Used: 0% You have currently used 1 asynchronous Apex operations out of an allowed 24-hour organization limit of 250,000. To learn about how this limit is calculated, see the Lightning Platform Apex Limits topic.' A table lists two jobs: one scheduled and one queueable.

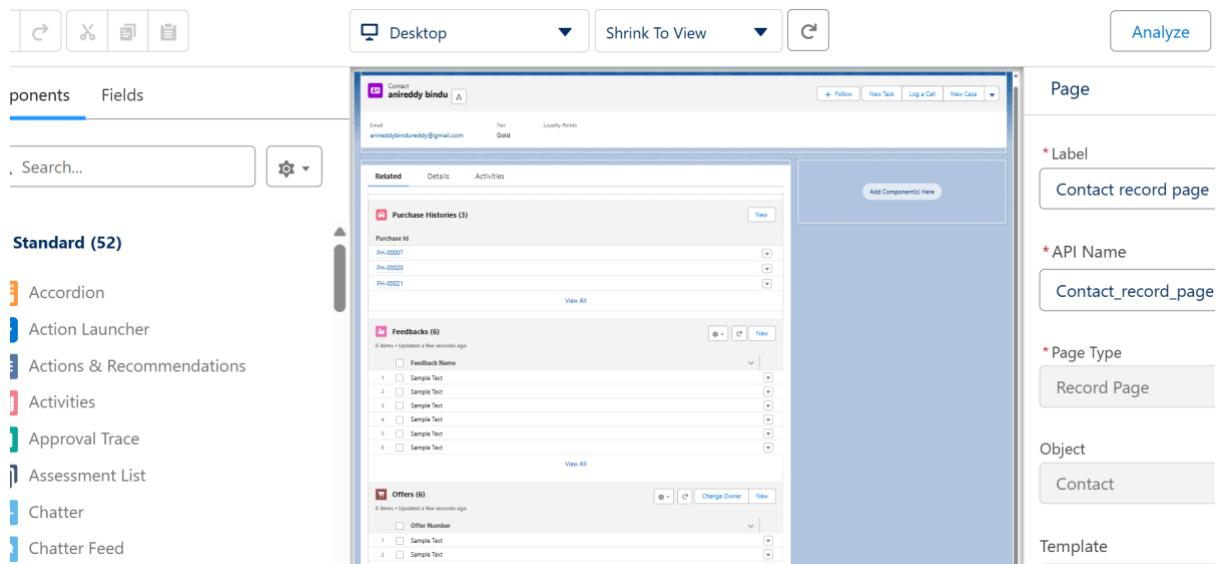
Action	Submitted Date	Job Type	Status	Status Detail	Total Batches	Batches Processed	Failures	Submitted By	Completion Date	Apex Class
	9/25/2025, 2:14 PM	Scheduled Apex	Queued		0	0	0	Anireddy, Bindu		RetailScheduled
	9/25/2025, 2:02 PM	Queueable	Completed		0	0	0	Anireddy, Bindu	9/25/2025, 2:02 PM	SimpleQueueable

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.



The screenshot shows the Lightning App Builder interface. On the left, a sidebar lists components like Accordion, Action Launcher, etc. The main area shows a contact record for 'anireddy bindu' with sections for Purchase Histories, Feedbacks, and Offers. To the right, configuration panels show settings for the 'Page', 'API Name' (Contact_record_page), 'Page Type' (Record Page), 'Object' (Contact), and 'Template'.

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

The screenshot shows a customer record in the Lightning App Builder. On the left, there are four tabs: 'Purchase Histories (3)', 'Feedbacks (6)', 'Offers (6)', and 'Contact Offers (1)'. Each tab displays a list of items with sample data. On the right, the 'Tabs' component configuration pane is open, showing the 'Label' as 'Tabs', the 'Default Tab' as 'Related', and three tabs defined: 'Related', 'Details', and 'Activities'. There are also sections for 'Add Tab' and 'Set Component Visibility'.

Home Page

Use Case:

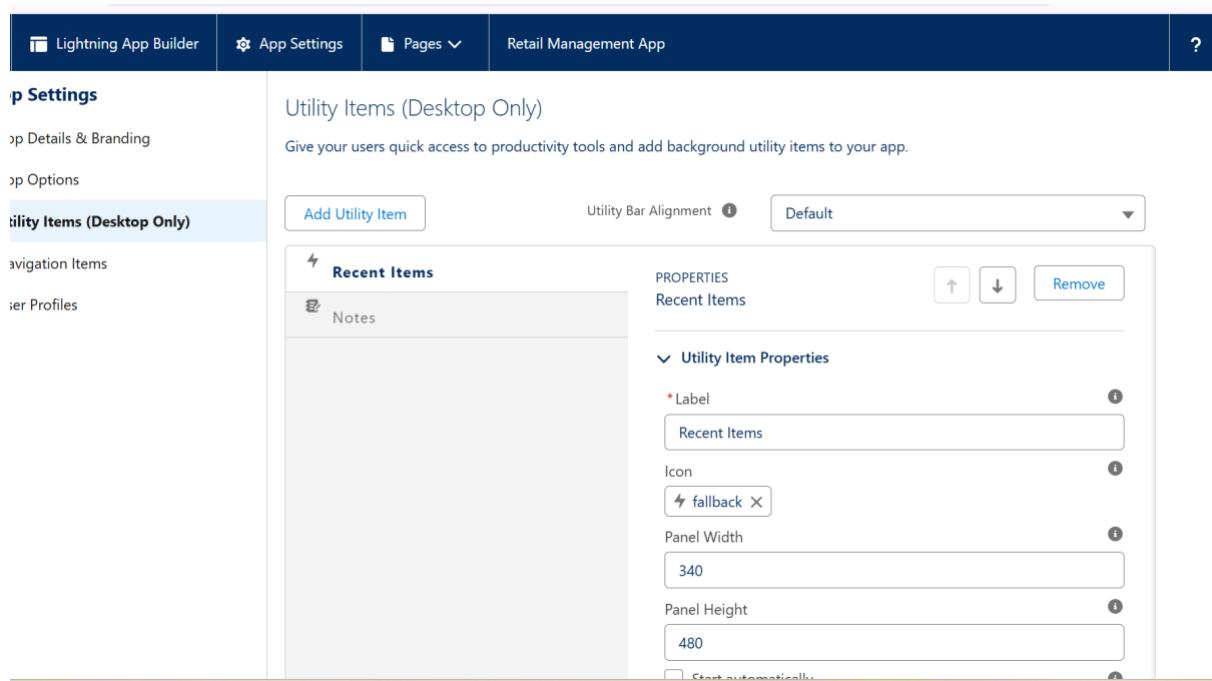
Provide sales reps and managers with a quick overview of key retail metrics, like pending orders, high-value customers, and sales performance, right when they log in.

The screenshot shows the configuration of a 'Home Page' component in the Lightning App Builder. The component preview shows a dashboard with sections for 'Lead conversion rate' and 'View Report'. The configuration pane on the right includes fields for 'Label' (set to 'Retail'), 'API Name' (set to 'Retail'), 'Page Type' (set to 'Home Page'), 'Template' (set to 'Home Template One Region'), and a 'Description' field.

Utility Bar

Use Case:

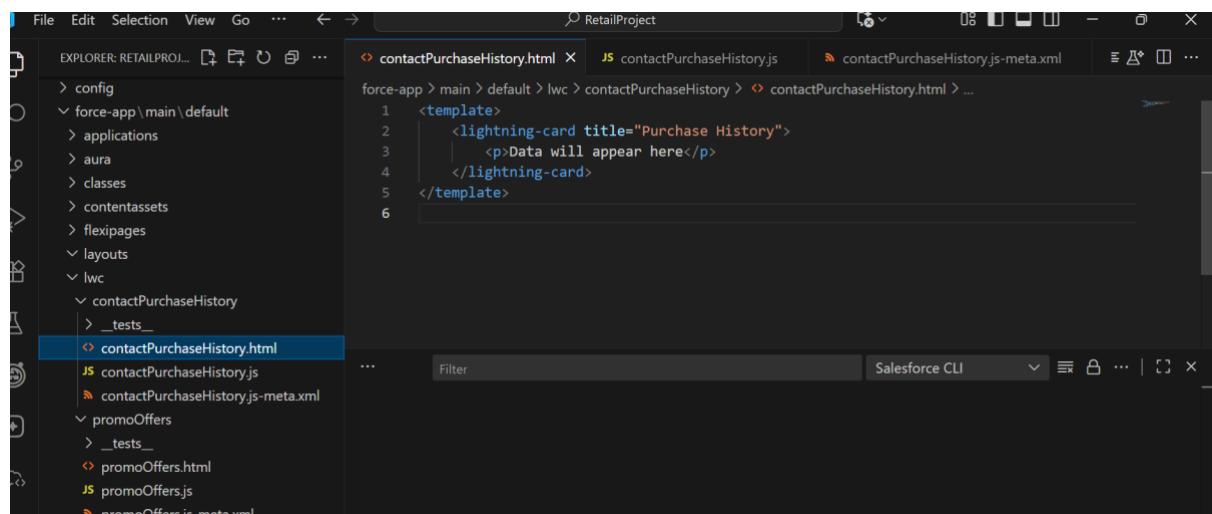
Provide sales reps with **quick access to notes, recent orders, or custom retail tools** without navigating away from the current record, improving efficiency during customer interactions.



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.



Page

* Label: Contact record page

* API Name: Contact_record_page

* Page Type: Record Page

Object: Contact

Template

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.

Highlights Panel in Header template region

Email: anireddybindureddy@gmail.com

Tier: Gold

Loyalty Points

Purchase Histories (3)

Purchase Id

- PH-00007
- PH-00020
- PH-00021

[View All](#)

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.

```

<template>
  <div class="promo-container">
    <h1>Current Promotions</h1>
    <ul>
      <template for:each={promotions} for:item="promo">
        <li key={promo.id}>
          {promo.name} - {promo.discount}%
        </li>
      </template>
    </ul>
  </div>
</template>

```

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.

The screenshot shows the 'SETUP > NAMED CREDENTIALS' interface. A 'SimulatedPOS' credential is selected. The page includes fields for Label ('SimulatedPOS'), Name ('SimulatedPOS'), URL ('https://example.com'), and 'Enabled for Callouts' (checkbox). It also features sections for 'Authentication' (set to 'Anonymous') and 'Callout Options'. A 'Generate Authorization Header' button is at the bottom.

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.

The screenshot shows the Salesforce Apex Classes setup page. At the top, there's a blue header bar with a gear icon and the word "SETUP". Below it, a blue bar says "Apex Classes". The main area has a title "Apex Class" and a "Help for this Page" link. A toolbar below the title includes "Apex Class Edit" and buttons for "Save", "Quick Save", and "Cancel". There are two tabs: "Apex Class" (which is selected) and "Version Settings". The main content area is a code editor with syntax highlighting for Apex. The code shown is:

```
1 @RestResource(urlMapping='/Purchases/*')
2 global with sharing class RetailPurchaseREST {
3
4     @HttpGet
5     global static Order__c getPurchaseById() {
6         RestRequest req = RestContext.request;
7         String orderId = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
8         return [SELECT Id, Name, Amount__c, Status__c FROM Order__c WHERE Id = :orderId LIMIT 1];
9     }
10 }
11
```

Callouts

Use Case:

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

The screenshot shows the Salesforce developer console. The top navigation bar includes "File", "Edit", "Debug", "Test", "Workspace", "Help", and tabs for "PurchaseHistoryTrigger.apxt", "PurchaseHistoryHandler.apxc", "OfferService.apxc", and "RetailCallout.apxc" (which is selected). Below the tabs, there are dropdowns for "Code Coverage: None" and "API Version: 64". The main content area is a code editor with syntax highlighting for Apex. The code shown is:

```
1 public class RetailCallout {
2
3     public static void getExchangeRate() {
4         Http http = new Http();
5         HttpRequest request = new HttpRequest();
6         request.setEndpoint('https://api.exchangerate-api.com/v4/latest/USD'); // example endpoint
7         request.setMethod('GET');
8
9         HttpResponse response = http.send(request);
10        System.debug('Response: ' + response.getBody());
11    }
12 }
```

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.

The screenshot shows the 'Platform Event Definition Detail' page for 'Order Placed'. The event is triggered when a new order is placed. It has a singular label 'Order Placed' and a plural label 'Orders Placed'. The object name is 'Order_Placed' and the API name is 'Order_Placed_e'. The event type is 'High Volume' and it publishes immediately. The created by field is 'Bindu Anireddy' and the modified by field is also 'Bindu Anireddy'. The deployment status is 'Deployed'.

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		

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.

The screenshot shows the 'Change Data Capture' setup page. It lists available entities on the left and selected entities on the right. Available entities include Account Clean Info, Account Contact Role, Agent Work, Asset, Asset Relationship, Assigned Resource, and Associated Location. Selected entities include Account, Contact, and Order.

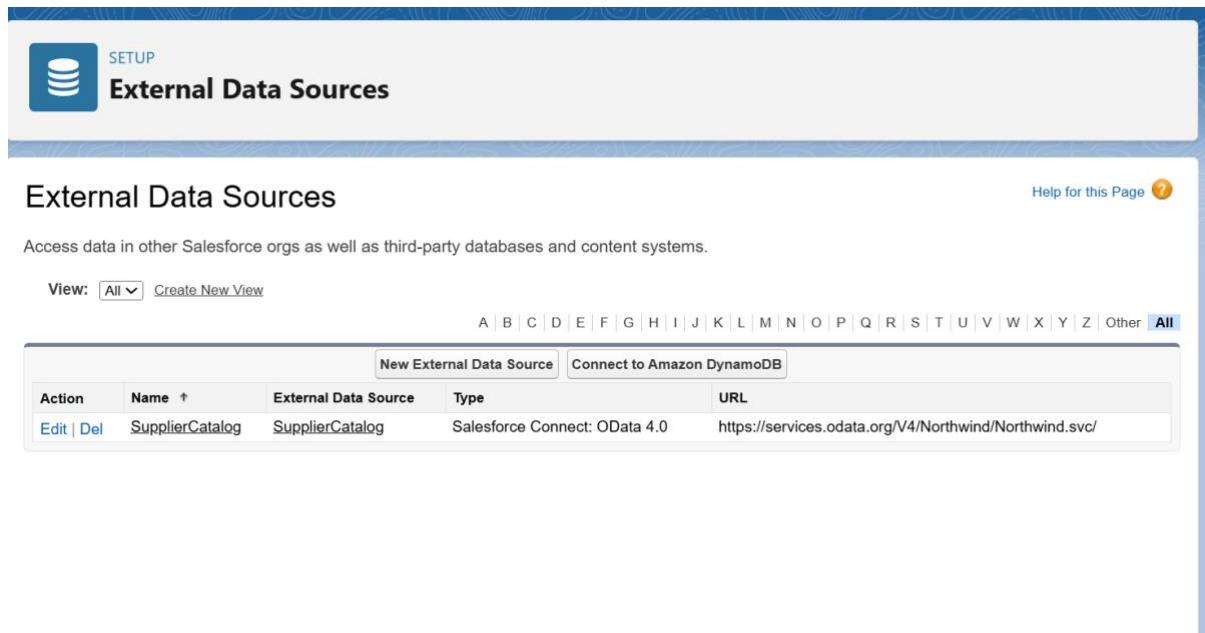
Available Entities	Selected Entities
Account Clean Info (AccountCleanInfo)	Account (Account)
Account Contact Role (AccountContactRole)	Contact (Contact)
Agent Work (AgentWork)	Order (Order_c)
Asset (Asset)	
Asset Relationship (AssetRelationship)	
Assigned Resource (AssignedResource)	
Associated Location (AssociatedLocation)	

Salesforce Connect

Use Case:

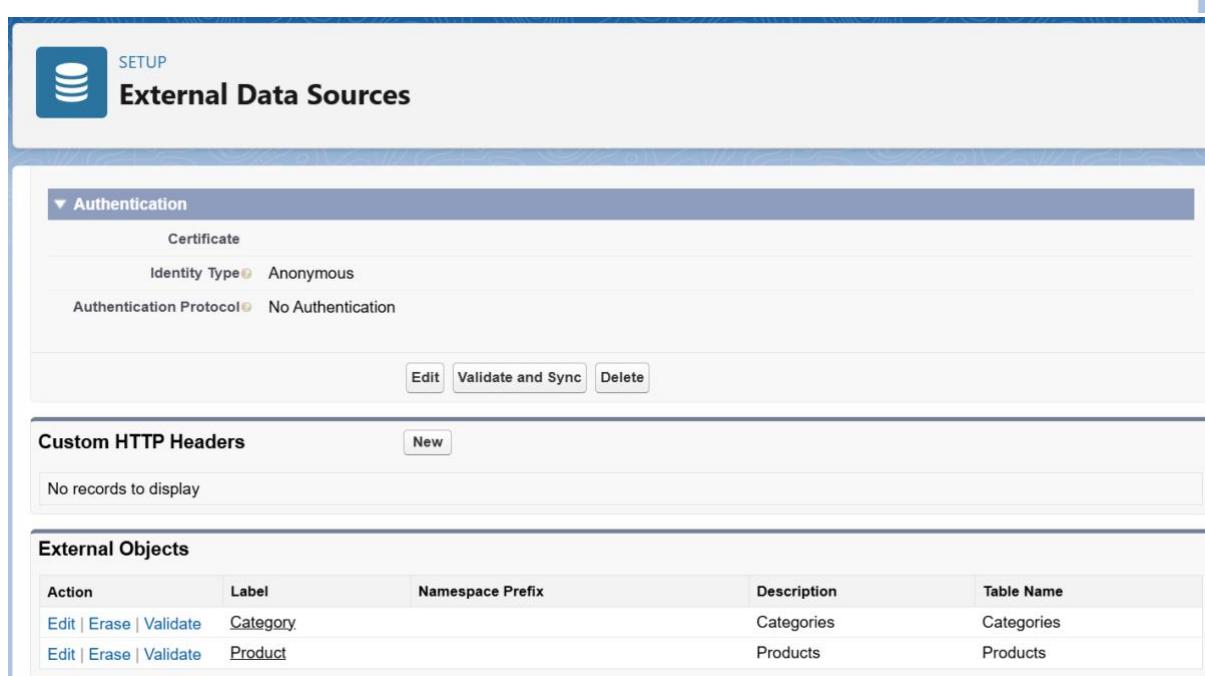
Access and display **external system data**, like inventory or supplier info, in Salesforce without storing it locally.

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



The screenshot shows the 'External Data Sources' setup page in Salesforce. At the top, there's a navigation bar with 'SETUP' and a 'External Data Sources' icon. Below the header, the title 'External Data Sources' is displayed, along with a 'Help for this Page' link. A sub-header 'External Data Sources' is followed by a descriptive text: 'Access data in other Salesforce orgs as well as third-party databases and content systems.' Below this, there are filtering options: 'View: All' and 'Create New View'. A navigation bar with letters A-Z and 'All' follows. A table lists one external data source:

Action	Name	External Data Source	Type	URL
Edit Del	SupplierCatalog	SupplierCatalog	Salesforce Connect: OData 4.0	https://services.odata.org/V4/Northwind/Northwind.svc/



The second part of the screenshot shows the 'Authentication' section of the 'External Data Sources' setup page. It includes fields for 'Identity Type' (set to 'Anonymous') and 'Authentication Protocol' (set to 'No Authentication'). Below these are buttons for 'Edit', 'Validate and Sync', and 'Delete'. A 'Custom HTTP Headers' section with a 'New' button is also present. At the bottom, there's a table for 'External Objects' with two entries:

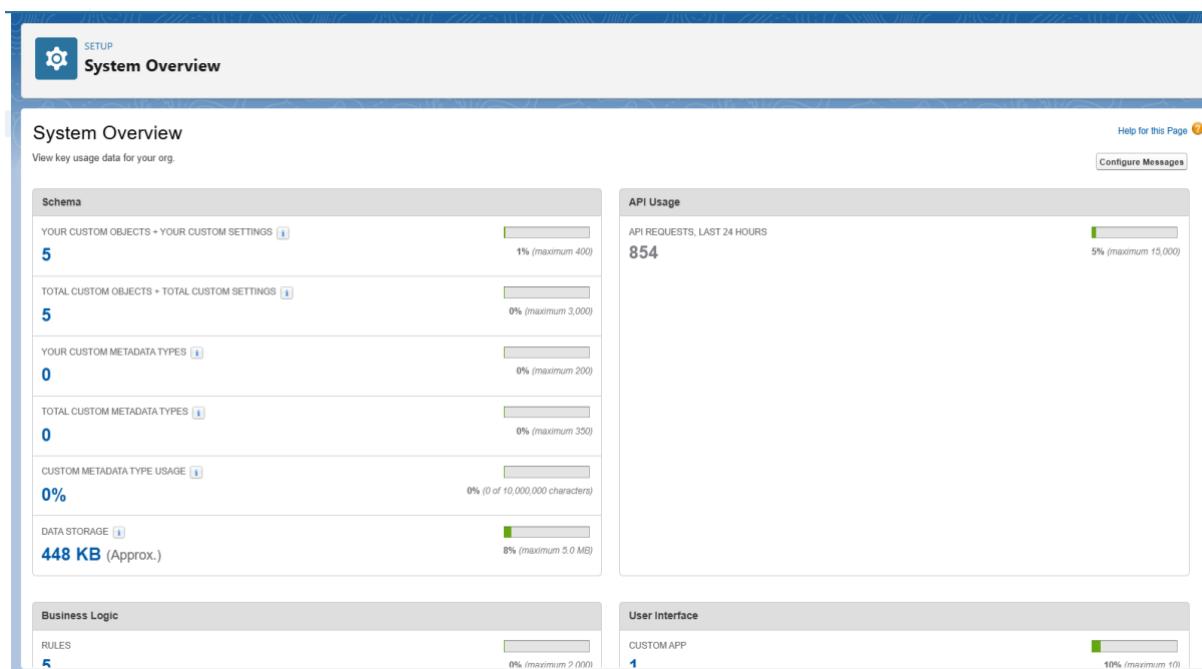
Action	Label	Namespace Prefix	Description	Table Name
Edit Erase Validate	Category		Categories	Categories
Edit Erase Validate	Product		Products	Products

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.

Remote Site Details

Remote Site Detail

Remote Site Name	ApexDevNet
Remote Site URL	http://www.apexdevnet.com
Disable Protocol Security	
Description	
Active	<input checked="" type="checkbox"/>
Created By	OrgFarm EPIC 9/15/2025, 7:42 AM

Modified By OrgFarm EPIC 9/15/2025, 7:42 AM

Edit Delete Clone

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.

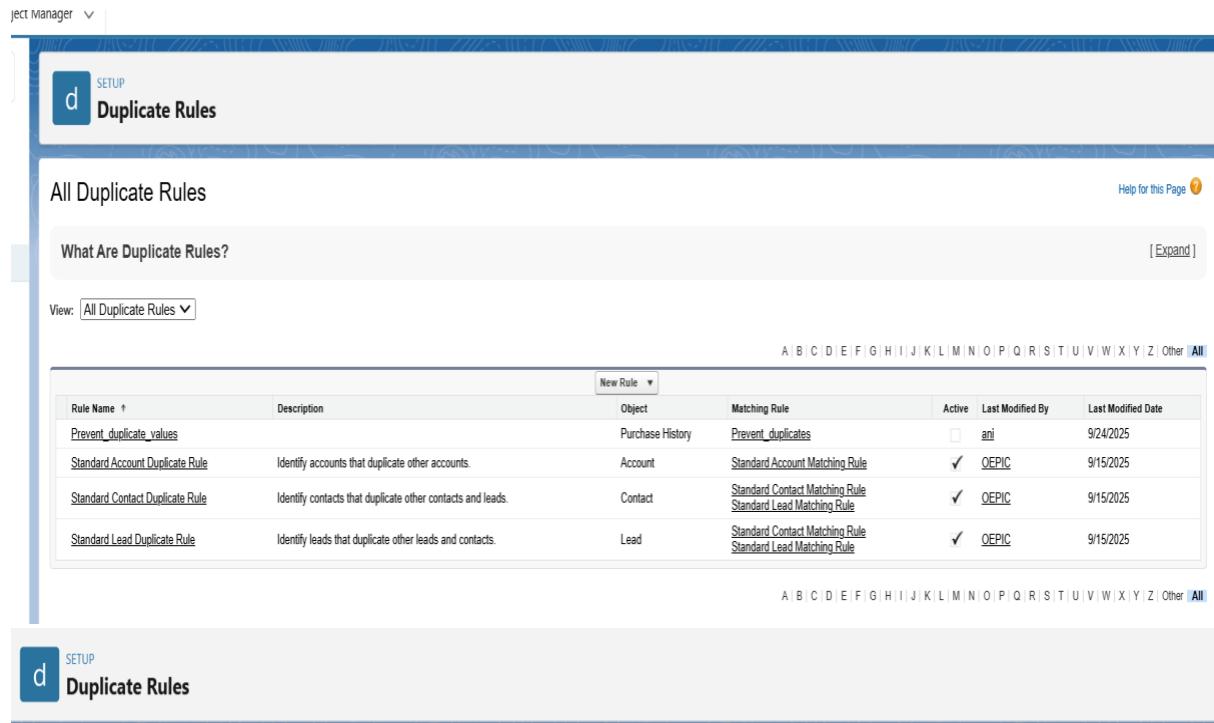
Ideal for **migrating large datasets** from legacy systems or external databases into Salesforce for retail operations.

Duplicate Rules

Use Case:

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

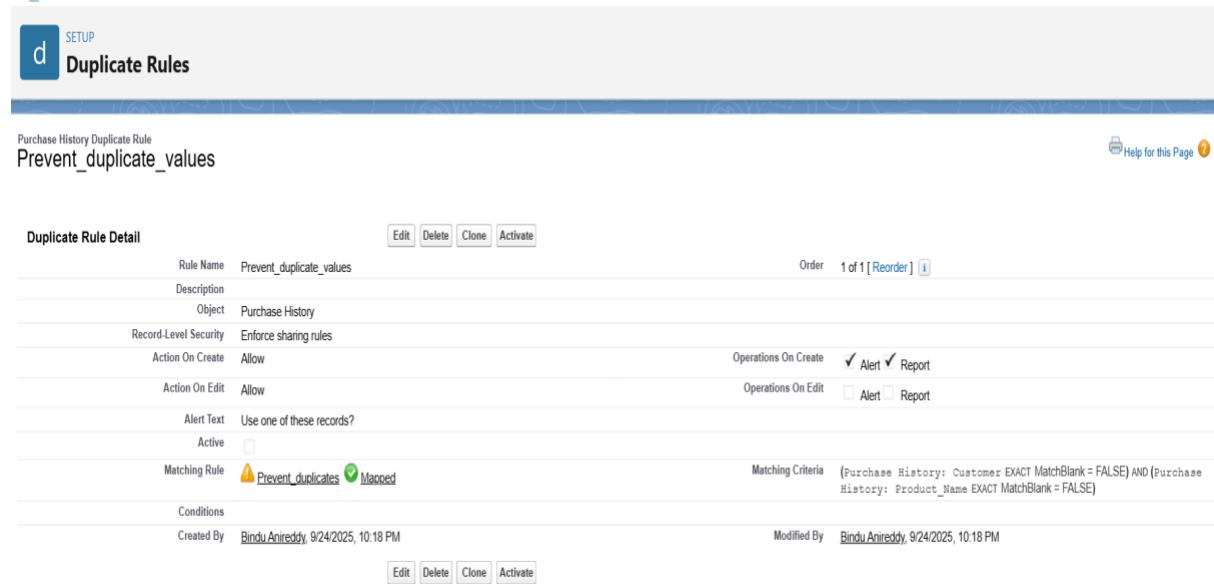
Helps sales reps avoid contacting the same customer multiple times and **maintains data integrity** for reporting and analytics.



The screenshot shows the 'All Duplicate Rules' page in the Salesforce Setup. The page title is 'd SETUP Duplicate Rules'. It includes a 'What Are Duplicate Rules?' section with a link to 'View: All Duplicate Rules'. A table lists four duplicate rules:

Rule Name	Description	Object	Matching Rule	Active	Last Modified By	Last Modified Date
Prevent_duplicate_values		Purchase History	Prevent_duplicates	<input type="checkbox"/>	ani	9/24/2025
Standard Account Duplicate Rule	Identify accounts that duplicate other accounts.	Account	Standard Account Matching Rule	<input checked="" type="checkbox"/>	OEPIC	9/15/2025
Standard Contact Duplicate Rule	Identify contacts that duplicate other contacts and leads.	Contact	Standard Contact Matching Rule Standard Lead Matching Rule	<input checked="" type="checkbox"/>	OEPIC	9/15/2025
Standard Lead Duplicate Rule	Identify leads that duplicate other leads and contacts.	Lead	Standard Contact Matching Rule Standard Lead Matching Rule	<input checked="" type="checkbox"/>	OEPIC	9/15/2025

Below the table, there is another table for the 'Purchase History Duplicate Rule' named 'Prevent_duplicate_values'.



The 'Duplicate Rule Detail' page for 'Prevent_duplicate_values' shows the following details:

Duplicate Rule Detail		Edit Delete Clone Activate	
Rule Name	Prevent_duplicate_values	Order	1 of 1 [Reorder]
Description			
Object	Purchase History		
Record-Level Security	Enforce sharing rules		
Action On Create	Allow	Operations On Create	<input checked="" type="checkbox"/> Alert <input checked="" type="checkbox"/> Report
Action On Edit	Allow	Operations On Edit	<input type="checkbox"/> Alert <input type="checkbox"/> Report
Alert Text	Use one of these records?		
Active	<input type="checkbox"/>		
Matching Rule	⚠ Prevent_duplicates Mapped	Matching Criteria	(Purchase_History: Customer EXACT MatchBlank = FALSE) AND (Purchase_History: Product_Name EXACT MatchBlank = FALSE)
Conditions			
Created By	Bindu Anireddy, 9/24/2025, 10:18 PM	Modified By	Bindu Anireddy, 9/24/2025, 10:18 PM

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.

The screenshot shows the 'Schedule Data Export' page under 'Data Export' in the 'Setup' menu. The page has two main sections: 'Schedule Data Export' and 'Exported Data'.
Schedule Data Export: This section includes fields for 'Export File Encoding' (ISO-8859-1), 'Include images, documents, and attachments' (unchecked), 'Include Salesforce Files and Salesforce CRM Content document versions' (unchecked), 'Replace carriage returns with spaces' (checked), and a 'Frequency' section. The frequency is set to 'On day 1 of every month'. The 'Start' date is 9/25/2025 and the 'End' date is 10/30/2025. A note below states: 'Exact start time will depend on job queue activity.'
Exported Data: This section allows selecting data types to include in the export. Checked items include Order, Account, OrderItem, Contact, and Opportunity. Unchecked items include All data, Contract, Asset, Lead, and Partner.

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.

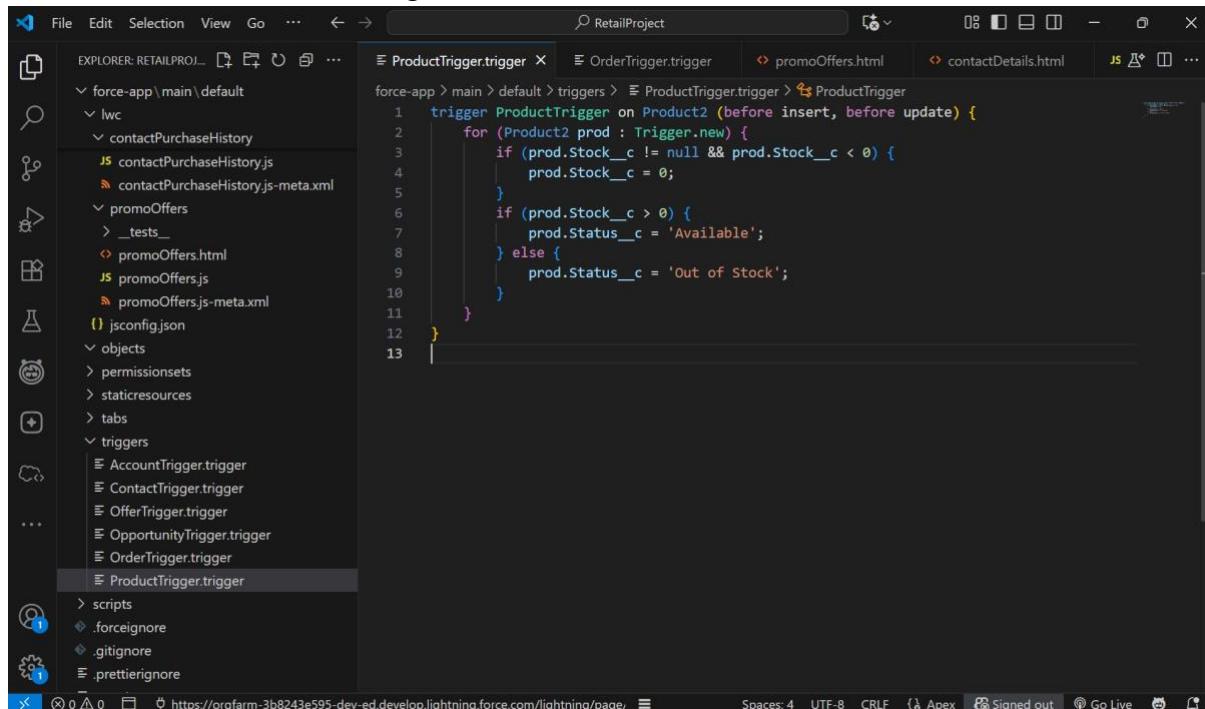
Ideal for version-controlled deployments and repetitive migrations from sandbox to production without using Change Sets.

VS Code & SFDX

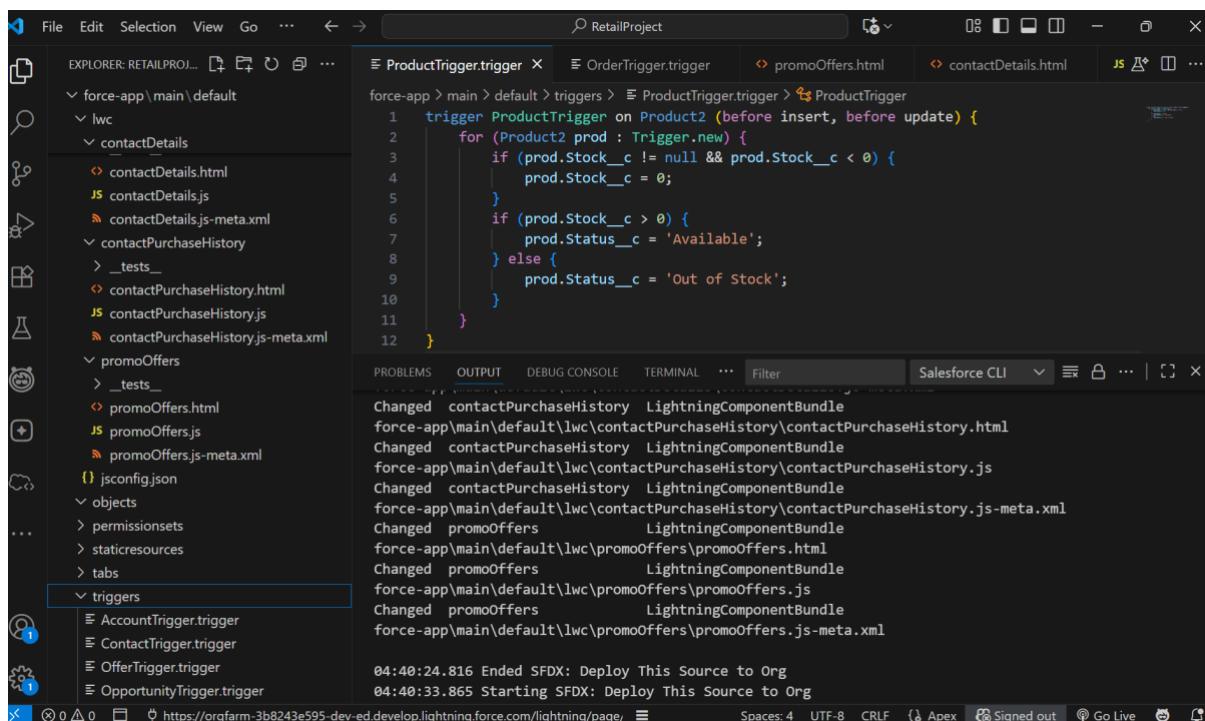
Use Case:

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.



```
force-app > main > default > triggers > ProductTrigger.trigger > ProductTrigger
1 trigger ProductTrigger on Product2 (before insert, before update) {
2     for (Product2 prod : Trigger.new) {
3         if (prod.Stock__c != null & prod.Stock__c < 0) {
4             prod.Stock__c = 0;
5         }
6         if (prod.Stock__c > 0) {
7             prod.Status__c = 'Available';
8         } else {
9             prod.Status__c = 'Out of Stock';
10        }
11    }
12 }
13 }
```



```
force-app > main > default > triggers > ProductTrigger.trigger > ProductTrigger
1 trigger ProductTrigger on Product2 (before insert, before update) {
2     for (Product2 prod : Trigger.new) {
3         if (prod.Stock__c != null & prod.Stock__c < 0) {
4             prod.Stock__c = 0;
5         }
6         if (prod.Stock__c > 0) {
7             prod.Status__c = 'Available';
8         } else {
9             prod.Status__c = 'Out of Stock';
10        }
11    }
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... Filter Salesforce CLI

```
Changed contactPurchaseHistory LightningComponentBundle
force-app\main\default\lwc\contactPurchaseHistory\contactPurchaseHistory.html
Changed contactPurchaseHistory LightningComponentBundle
force-app\main\default\lwc\contactPurchaseHistory\contactPurchaseHistory.js
Changed contactPurchaseHistory LightningComponentBundle
force-app\main\default\lwc\contactPurchaseHistory\contactPurchaseHistory.js-meta.xml
Changed promoOffers LightningComponentBundle
force-app\main\default\lwc\promoOffers\promoOffers.html
Changed promoOffers LightningComponentBundle
force-app\main\default\lwc\promoOffers\promoOffers.js
Changed promoOffers LightningComponentBundle
force-app\main\default\lwc\promoOffers\promoOffers.js-meta.xml

04:40:24.816 Ended SFDX: Deploy This Source to Org
04:40:33.865 Starting SFDX: Deploy This Source to Org
```

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.

The screenshot shows a report interface with a header bar containing navigation links: Sales, Home, Opportunities, Leads, Tasks, Files, Accounts, Contacts, and Campaigns. Below the header is a title bar with a document icon and the text "Report: Purchase History History" followed by "Copy of New Purchase History History Rep". The main content area displays a summary table with two rows: "Total Records" (7) and "Total Amount" (\$13,490.00). Below this is a detailed table with 8 rows of purchase history data, including columns for Purchase Id, Payment Method, Product Name, Amount, and Status. The total amount for all purchases is listed at the bottom of the table as \$13,490.00.

	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.

The screenshot shows the "Custom Report Types" setup screen. At the top, there is a "Purchase Report" section with a note about previewing or updating information. To the right are buttons for "Preview Layout", "Edit Layout", "Clone", "Delete", and "Close". Below this are two main sections: "Details" and "Object Relationships". The "Details" section contains fields for Display Label (Purchase Report), API Name (Purchase_Report), Description (Purchase History report based on purchased items), Created By (Bindu Anireddy, 9/26/25, 5:03 AM), Store in Category (accounts), Deployment Status (In Development), and Modified By (Bindu Anireddy, 9/26/25, 5:03 AM). The "Object Relationships" section shows a diagram illustrating the relationship between "Purchase Histories (A)" and "Feedbacks (B)", stating that at least one related record from Feedbacks (B) is required. A legend below the diagram identifies "A" as blue and "B" as orange.

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.

The screenshot shows a Salesforce dashboard interface. At the top, there's a navigation bar with links for Sales, Home, Opportunities, Leads, Tasks, Files, Accounts, Contacts, Campaigns, Dashboards (which is currently selected), Reports, Chatter, and Groups. Below the navigation bar, there's a search bar with placeholder text "All Sales Overview". Underneath the search bar, there are two main sections: "Purchase History History Report" and "Copy of New Purchase History History Rep". Both sections contain tables with columns for Purchase History ID, Purchase History Purchase Id, Amount, Edit Date, Payment Method, Product Name, and Status. The "Purchase History History Report" table has 7 rows of data, while the "Copy of New Purchase History History Rep" table has 7 rows of data. At the bottom of the dashboard, there are buttons for "Report (New Purchase History History Report)" and "View Report (Copy of New Purchase History History Rep)".

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.

The screenshot shows the "Profiles" section of the Salesforce setup. At the top, there's a header with a user icon, "SETUP", and "Profiles". Below the header, there's a sub-header "Profile" and the name "Customer Manager". A note says "Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information." Another note says "If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile." Below these notes, there's a "Profile Detail" section with fields for Name (Customer Manager), User License (Salesforce Platform), Description, Created By (Bindu Anireddy, 9/25/2025, 6:24 AM), Modified By (Bindu Anireddy, 9/25/2025, 1:57 PM), and a "Custom Profile" checkbox which is checked. At the bottom, there's a "Page Layouts" section titled "Standard Object Layouts" with a table mapping objects to their respective layouts. For example, Global has "Global Layout [View Assignment]", Email Application has "Not Assigned [View Assignment]", Home Page Layout has "Home Page Default [View Assignment]", Account has "Account Layout [View Assignment]", Alternative Payment Method has "Alternative Payment Method Layout [View Assignment]", Lead has "Lead Layout [View Assignment]", Location has "Location Layout [View Assignment]", Location Group has "Location Group Layout [View Assignment]", Location Group Assignment has "Location Group Assignment Layout [View Assignment]", and Object Milestone has "Object Milestone Layout [View Assignment]".

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.

The screenshot shows the 'Roles' page in the Salesforce Setup. At the top, there's a header with a user icon and the word 'SETUP'. Below it is a section titled 'Your Organization's Role Hierarchy' with a 'Collapse All' and 'Expand All' button. The hierarchy tree is as follows:

- Retail Loyalty CRM Project
 - Add Role
 - CEO** Edit | Del | Assign
 - Add Role
 - CFO** Edit | Del | Assign
 - Add Role
 - COO** Edit | Del | Assign
 - Add Role
 - Marketing Manager** Edit | Del | Assign
 - Add Role
 - Marketing Analyst** Edit | Del | Assign
 - Add Role
 - Sales Manager** Edit | Del | Assign
 - Add Role
 - Sales Rep** Edit | Del | Assign
 - Add Role
 - SVP, Customer Service & Support** Edit | Del | Assign
 - Add Role
 - Customer Support, International** Edit | Del | Assign
 - Add Role
 - Customer Support - North America** Edit | Del | Assign

Users

Use Case:

Create and manage **individual accounts** for employees.

Example: Add a **new cashier** as a Salesforce User with limited retail app access.

The screenshot shows the 'Users' page in the Salesforce Setup. At the top, there's a header with a user icon and the word 'SETUP'. Below it is a section titled 'All Users' with a 'Help for this Page' link. The page displays a list of users with columns for Action, Full Name, Alias, Username, Role, Active, and Profile. The users listed are:

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/>	Agent Service	sracegard	binduadv@domain.com	Marketing Analyst	✓	Standard Platform User
<input type="checkbox"/> Edit	Anirudey Bindu	ani	anirudeybindu217@agentforce.com	CEO	✓	System Administrator
<input type="checkbox"/>	Chatter Expert	Chatter	chatty_00dg@000000mewzuan.sph205sdomz@chatter.salesforce.com		✓	Chatter Free User
<input type="checkbox"/> Edit	EPIC_OrgFarm	OEPIIC	epic_4b124a3674c@orgfarm.salesforce.com	Sales Rep	✓	System Administrator
<input type="checkbox"/> Edit	Manager_Store	storemgr	bindukuntemgr@xarnele.com	Sales Manager	✓	Standard User
<input type="checkbox"/> Edit	User_Integration	integ	integration@00dg@000000mewzuan.com		✓	Analytics Cloud Integration User
<input type="checkbox"/> Edit	User_Marketing	mktguser	bindukuntk@domain.com	Marketing Manager	✓	Standard User
<input type="checkbox"/> Edit	User_Security	sec	insightssecurity@00dg@000000mewzuan.com		✓	Analytics Cloud Security User

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.

Permission Set Overview

Description: "Grants Create/Edit/Delete on Purchase History object"

License: Salesforce

Session Activation Required:

Permission Set Groups Added To: 0

API Name: Purchase_History_Editor

Namespace Prefix: BinduAnireddy

Created By: Bindu Anireddy, 9/25/2025, 7:14 AM

Last Modified By: Bindu Anireddy, 9/25/2025, 7:23 AM

Assigned Apps

Assigned Connected Apps

Object Settings

App Permissions

Apex Class Access

Visualforce Page Access

External Data Source Access

Flow Access

Named Credential Access

External Credential Principal Access

Learn More

007j9mk%3Fisdt%3Dp1%27)

Object Manager

Tab Settings

Available Visible

Object Permissions

Permission Name	Enabled
Read	<input checked="" type="checkbox"/>
Create	<input checked="" type="checkbox"/>
Edit	<input checked="" type="checkbox"/>
Delete	<input checked="" type="checkbox"/>
View All Records	<input checked="" type="checkbox"/>
Modify All Records	<input checked="" type="checkbox"/>
View All Fields	<input checked="" type="checkbox"/>

Field Permissions

Field Name	Field API Name	Read Access	Edit Access
Amount	Amount_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Created By	CreatedById	<input type="checkbox"/>	<input type="checkbox"/>
Customer	Customer_c	<input type="checkbox"/>	<input type="checkbox"/>
Customer Account	Customer_Account_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Discount Percentage	Discount_Percentage_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Last Modified By	LastModifiedById	<input type="checkbox"/>	<input type="checkbox"/>
Loyalty Points	Loyalty_Points_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Payment Method	Payment_Method_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Points Earned	Points_Earned_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Product Name	Product_Name_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Purchase Date	Purchase_Date_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Purchase Id	Name	<input type="checkbox"/>	<input type="checkbox"/>
Quantity	Quantity_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sales Rep	Sales_Rep_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
SKU	SKU_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Status	Status_c	<input type="checkbox"/>	<input type="checkbox"/>
Store Location	Store_Location_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Tier	Tier_c	<input checked="" type="checkbox"/>	<input type="checkbox"/>

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	✓	
<hr/>				
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.

Setup Offer Sharing Rule

Use sharing rules to make automatic exceptions to your organization-wide sharing settings for defined sets of users.

Note: "Roles and subordinates" includes all users in a role, and the roles below that role. This includes portal roles that may give access to users outside the organization.

You can use sharing rules only to grant wider access to data, not to restrict access.

Label	offer rule
Rule Name	offer_rule
Description	
Step 1: Select your rule type	
Criteria	Field Operator Value AND
	Discount Percentage greater or equal 20 AND
	None None AND
	None None AND
	None None AND
	None None
Add Filter Logic	
Additional Options	<input checked="" type="checkbox"/> Include records owned by users who can't have an assigned role
Share with	All Internal Users
Access Level	Read/Write
Created By	Bindu Anireddy, 9/25/2025, 7:35 AM
Modified By Bindu Anireddy, 9/25/2025, 7:35 AM	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

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.

The screenshot shows the 'Sharing Settings' page in Salesforce. At the top, there's a header with a 'SETUP' button and a 'Sharing Settings' title. Below the header, a section titled 'Sharing Settings' is described as displaying organization-wide sharing settings. It includes a dropdown for 'Manage sharing settings for: Purchase History'. There are sections for 'Default Sharing Settings' (Organization-Wide Defaults and Other Settings), 'Sharing Rules' (disabled), and 'Sharing Overrides' (Profiles That Override Parent's Sharing). A note at the bottom states: 'Organization-wide permissions affect all objects in the organization. Object permissions affect only the given object.'

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.

The screenshot shows the 'Customer Account' field-level security configuration. It includes a table where rows represent users or profiles and columns indicate whether the field is visible or read-only.

	Visible	Read-Only
Analytics Cloud Integration User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Analytics Cloud Security User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Anypoint Integration	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Authenticated Website	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Website	<input type="checkbox"/>	<input type="checkbox"/>
B2B Reordering Portal Buyer Profile	<input type="checkbox"/>	<input type="checkbox"/>
Contract Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Cross Org Data Proxy User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Marketing Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Sales Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Support Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Customer Community Login User	<input type="checkbox"/>	<input type="checkbox"/>
Customer Community Plus Login User	<input type="checkbox"/>	<input type="checkbox"/>
Customer Community Plus User	<input type="checkbox"/>	<input type="checkbox"/>
Customer Community User	<input type="checkbox"/>	<input type="checkbox"/>

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.

The screenshot shows the 'Session Settings' configuration page. It includes sections for Session Timeout, Session Settings, Extended use of IE11 with Lightning Experience, and Caching.

Session Timeout: Set to 2 hours. Options include Disable session timeout warning popup (unchecked) and Force logout on session timeout (checked).

Session Settings: Includes checkboxes for Lock sessions to the IP address from which they originated (unchecked), Lock sessions to the domain in which they were first used (checked), Terminate all of a user's sessions when an admin resets that user's password (unchecked), Force relogin after Login-As-User (checked), Require HttpOnly attribute (unchecked), Use POST requests for cross-domain sessions (unchecked), Enforce login IP ranges on every request (unchecked), and When embedding a Lightning application in a third-party site, use a session token instead of a session cookie (unchecked).

Extended use of IE11 with Lightning Experience: A note states: "EXTENDED USE OF IE11 WITH LIGHTNING EXPERIENCE HAS NOW ENDED** AS OF DECEMBER 31, THE EXTENDED PERIOD HAS ENDED, AND USE OF INTERNET EXPLORER 11 (IE 11) WITH LIGHTNING EXPERIENCE IS NO LONGER SUPPORTED. ISSUES WITH PERFORMANCE OR FUNCTIONALITY THAT AFFECT ONLY IE 11 WILL NOT BE FIXED. PLEASE SWITCH TO A SUPPORTED BROWSER."

Caching: Includes checkboxes for Enable caching and autocomplete on login page (checked), Enable secure and persistent browser caching to improve performance (checked), Enable user switching (checked), Remember me until logout (unchecked), and Enable Content Delivery Network (CDN) for Lightning Component framework (checked).

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.

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).

Date	User	Source Namespace Prefix	Action	Section	Delegate User
9/25/2025, 4:10:40 PM PDT	anireddybindurreddy217@agentforce.com		Changed promoOffers Lightning Web Component	Lightning Components	
9/25/2025, 4:10:40 PM PDT	anireddybindurreddy217@agentforce.com		Changed contactPurchaseHistory Lightning Web Component	Lightning Components	
9/25/2025, 4:10:40 PM PDT	anireddybindurreddy217@agentforce.com		Changed contactDetails Lightning Web Component	Lightning Components	
9/25/2025, 3:35:15 PM PDT	anireddybindurreddy217@agentforce.com		Added Change Data Capture entity Order_c to channel ChangeEvents	Change Data Capture	
9/25/2025, 3:34:34 PM PDT	anireddybindurreddy217@agentforce.com		Added Change Data Capture entity Contact to channel ChangeEvents	Change Data Capture	
9/25/2025, 3:34:34 PM PDT	anireddybindurreddy217@agentforce.com		Added Change Data Capture entity Account to channel ChangeEvents	Change Data Capture	
9/25/2025, 3:29:50 PM PDT	anireddybindurreddy217@agentforce.com		Changed RetailCallout Apex Class code	Apex Class	
9/25/2025, 3:29:42 PM PDT	anireddybindurreddy217@agentforce.com		Created RetailCallout Apex Class code	Apex Class	
9/25/2025, 3:11:44 PM PDT	anireddybindurreddy217@agentforce.com		Changed contactDetails Lightning Web Component	Lightning Components	
9/25/2025, 3:10:34 PM PDT	anireddybindurreddy217@agentforce.com		Created contactDetails Lightning Web Component	Lightning Components	
9/25/2025, 2:57:01 PM PDT	anireddybindurreddy217@agentforce.com		Changed Lightning Page: Retail Management App UtilityBar	Lightning Pages	
9/25/2025, 2:49:44 PM PDT	anireddybindurreddy217@agentforce.com		Created custom app Retail Management App	Custom Apps	
9/25/2025, 2:49:44 PM PDT	anireddybindurreddy217@agentforce.com		Created Lightning Page: Retail Management App UtilityBar	Lightning Pages	
9/25/2025, 2:28:52 PM PDT	anireddybindurreddy217@agentforce.com		Created RetailOrderTest Apex Class code	Apex Class	
9/25/2025, 2:24:38 PM PDT	anireddybindurreddy217@agentforce.com		Changed RetailExceptionDemo Apex Class code	Apex Class	
9/25/2025, 2:22:35 PM PDT	anireddybindurreddy217@agentforce.com		Created RetailException Apex Class code	Apex Class	
9/25/2025, 2:19:45 PM PDT	anireddybindurreddy217@agentforce.com		Changed RetailFutureJob Apex Class code	Apex Class	
9/25/2025, 2:19:37 PM PDT	anireddybindurreddy217@agentforce.com		Created RetailFutureJob Apex Class code	Apex Class	
9/25/2025, 2:12:34 PM PDT	anireddybindurreddy217@agentforce.com		Changed RetailScheduledJob Apex Class code	Apex Class	
9/25/2025, 2:12:23 PM PDT	anireddybindurreddy217@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

New Contact

* = Required Information

Contact Information

Contact Owner  Bindu Anireddy	Phone <input type="text"/>
Account Name <input type="text" value="Search Accounts..."/> 	Home Phone <input type="text"/>
Title <input type="text"/>	Mobile <input type="text"/>
Department <input type="text"/>	Other Phone <input type="text"/>
Birthdate <input type="text"/> 	Fax <input type="text"/>
Reports To <input type="text" value="Search Contacts..."/> 	Assistant <input type="text"/>
Lead Source <input type="text" value="--None--"/>	Asst. Phone <input type="text"/>
Total Spent \$0.00 <i>This field is calculated upon save</i>	

Contact
Ms. Vaishnavi

Email vaish@example.com	Tier Silver	Loyalty Points 10
----------------------------	----------------	----------------------

Related **Details** **Activities**

-  **Purchase Histories (0)** 
-  **Feedbacks (0)** 
-  **Offers (0)**  
-  **Contact Offers (0)** 

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.

New Purchase History

* = Required Information

Information	
Purchase Id	
* Customer	Goutham
Purchase Date	9/24/2025
Product Name	Trouser
SKU	1
Quantity	3
Amount	-\$90.00
Store Location	Store: Bangalore
Payment Method	--None--
Points Earned	
<input type="button" value="Cancel"/> <input type="button" value="Save & New"/> <input type="button" value="Save"/>	

New Purchase History

* = Required Information

Information

Purchase Id

* Customer	Goutham
Purchase Date	9/24/2025
Product Name	Trouser
SKU	3
Quantity	3
Amount	<input type="text" value="-\$90.00"/>

Purchase Amount cannot be negative.

We hit a snag.

Review the following fields

- Amount

Store Location
Store: Hyderabad

Payment Method
UPI

Cancel Save & New Save

Points Earned

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.

New Purchase History

* = Required Information

Information

Purchase Id

* Customer

 anireddy bindu

Purchase Date



Product Name

SKU

Quantity

Amount

500



Store Location

--None--



Payment Method

--None--



Points Earned



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.

New Feedback: Praise

* = Required Information

Information

Feedback Name

* Customer
anireddy bindu

Purchase
PH-00030

Rating ⓘ
10

Comments
Good

Source
--None--

Sentiment
--None--

Submitted Date

Date Time

Status
--None--

Cancel Save & New Save

New Feedback: Praise

* = Required Information

Information

Feedback Name

* Customer
anireddy bindu

Purchase
PH-00030

Rating ⓘ
∅ 10

Rating: value outside of valid range on numeric field: 10

Comments
Good

Source
In-Store

Sentiment
Positive

Submitted Date
Date
9/23/2025

⚡ We hit a snag.

Review the following fields

- Rating

Status
Open

✖ Cancel Save & New Save

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:
 - o Assignment element: Calculate loyalty points = Purchase_Amount_c / 100 * 10
 - o Decision element: Map points to tier
 - e.g.,
 - 0–499 points → Silver
 - 500–999 points → Gold
 - 1000+ points → Platinum
 - o Update Records element: Update Customer Tier

3. Create a new Purchase History record:

- o Customer: Madhu
- o 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:
 - o Total Points = 500 (or cumulative total)
 - o Tier = Gold

New Purchase History

* = Required Information

Information	
Purchase Id	
* Customer	MAdhu
Purchase Date	9/16/2025
Product Name	
SKU	
Quantity	
Amount	5000
Store Location	--None--
Payment Method	--None--
Points Earned	
<input type="button" value="Cancel"/> <input type="button" value="Save & New"/> <input type="button" value="Save"/>	

Sales Home Opportunities Leads Tasks Files Accounts Contacts Campaigns Dashboards Reports Chat

 Purchase History
PH-00031

SKU	
Quantity	
Amount	\$5,000.00
Store Location	
Payment Method	
Points Earned	
Customer Account 	
Loyalty Points	598.00
Tier	Gold
Discount Percentage	0.00%
Status	
Sales Rep	
Created By  Bindu Anireddy, 9/25/2025, 6:32 PM	Last Modified By  Bindu Anireddy, 9/25/2025, 6:33 PM

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	Total Amount				
6	\$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:
 - o Field: Discount__c
 - o Condition: Greater than 10
4. Ensure the data source (report) has Purchase History records with various discount values:
 - o Purchase 1: Discount = 5 → should not appear
 - o Purchase 2: Discount = 12 → should appear
 - o 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:

Purchase Date ↑	Customer Account	Product Name	Amount	Status
-	Siri pannala	Kurti	\$800.00	Completed
9/12/2025	Bindu Reddy	Duppata	\$1.50k	Completed
9/23/2025	Goutham	Trouser	\$3.00k	Completed
9/24/2025	Siri pannala	jeans	\$5.00k	Completed
9/25/2025	Bindu Reddy	Kurti	\$2.00k	Completed
9/26/2025	-	Kurti	\$700.00	Completed

The screenshot shows a Salesforce dashboard titled "Purchase objects". At the top, it displays the date "As of Sep 25, 2025, 7:04 PM" and the user "Viewing as Bindu Anireddy". Below this are two filter sections: "Discount Percentage" set to "greater or equal 10" and "Payment Method" set to "All". A main report section titled "purchase history" contains a table with one row of data. The table columns are "Purchase Date ↑", "Customer Account", "Product Name", "Amount", and "Status". The data row shows: Purchase Date 9/12/2025, Customer Account Bindu Reddy, Product Name Duppata, Amount \$1.50k, and Status Completed. At the bottom of the report section, there are links "View Report (purchase history)" and "As of Sep 25, 2025, 7:04 PM".

Purchase Date ↑	Customer Account	Product Name	Amount	Status
9/12/2025	Bindu Reddy	Duppata	\$1.50k	Completed

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 simplifies 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.