FinSmart CRM – Loan & Customer Relationship Management System

Industry: Banking & Financial Services

Target Users: Loan Officers, Customers, Risk Analysts, and Bank Managers

Overview:

This project focuses on building a Salesforce-based **FinSmart CRM** solution to improve the bank's loan management process. The system will automate loan application intake, verification, and approval to reduce delays and manual errors. Customers will get a self-service portal to apply for loans, track their status, and receive automatic reminders for repayment schedules and overdue amounts. Integration with credit bureaus will allow instant credit score checks and personalized loan offers. Managers will benefit from real-time dashboards and reports that provide insights into loan portfolio health, repayment performance, risks, and revenue growth. Overall, the project aims to increase efficiency, reduce defaults, improve customer satisfaction, and support smarter decision-making.

Problem Statement:

A leading bank receives thousands of loan applications every month across different channels such as branch offices, online forms, and partner agencies. However, the loan approval and repayment management process faces multiple challenges:

- Loan application intake and approvals are **manual and time-consuming**, leading to delays.
- Repayment schedules are **not tracked efficiently**, causing high default risks.
- Customers do not receive **personalized offers** based on their credit history and past transactions.
- Managers lack **real-time visibility** into portfolio risk, loan performance, and defaulter trends.

To address these issues, the bank wants to implement a **Salesforce-based FinSmart CRM** that will include following **Objective:**

- Automate **loan application intake, verification, and approval workflows**.
- Provide **customers with a portal** to apply for loans and track their status.
- Send **automated reminders** for repayment schedules and overdue accounts.
- Integrate with **credit bureau APIs** for credit score checks.
- Deliver **dashboards and reports** on loan portfolio health, repayment performance, and revenue growth.

📄 Phase 1: Problem Understanding & Industry Analysis

1. Requirement Gathering

The banking institution currently faces multiple operational challenges:

- Manual loan application intake through branches and online forms results in **delays in approval processing**.
- Repayment schedules are maintained in **separate systems**, causing **missed follow-ups and defaults**.
- Customers do not receive **personalized loan offers**, limiting engagement.
- Managers lack **real-time visibility** into loan portfolio performance and risks.

The key requirements identified are:

- Automate **loan intake and approval workflows**.
- Provide **customer self-service portals** for loan tracking and document uploads.
- Automate **repayment reminders and overdue alerts**.
- Integrate with **credit bureau APIs** to validate credit scores.
- Create **dashboards** for loan performance, defaults, and revenue tracking.

2. Stakeholder Analysis

- **Customers (Borrowers):** Need a transparent and fast loan application process, repayment reminders, and personalized offers.
- **Loan Officers:** Require tools to manage loan applications, verify documents, and handle escalations efficiently.
- **Risk Analysts:** Need insights into credit history, defaulter trends, and portfolio risks.
- **Managers:** Require real-time dashboards on loan approvals, repayments, defaults, and revenue growth.
- **System Admin/Developers:** Need clear data models, automation, and integrations to ensure smooth operations.

3. Business Process Mapping

Current Process:
Loan Application (manual form) → Document Verification (offline) → Credit Check (delayed) → Manager Approval → Disbursement → Repayment Tracking (manual).

Proposed Salesforce-Enabled Process:
Loan Application (online portal) → Automated Document Verification → Real-time Credit Bureau Check → Automated Approval Workflow → Loan Disbursement → Repayment Reminders (Flows/Email/SMS) → Portfolio Dashboards.

4. Industry-specific Use Case Analysis

In the **Banking & Financial Services** industry, digital loan management systems are standard for improving efficiency. Competitors use advanced CRM solutions for:

- **Faster loan approvals** through automation.
- **Lower default rates** with repayment reminders.
- **Higher customer satisfaction** via self-service portals.
- **Cross-selling opportunities** using personalized offers.

FinSmart CRM will bring these industry best practices into the bank's operations.

5. AppExchange Exploration

Relevant Salesforce AppExchange apps to consider:

- **Credit Bureau Integration Apps** (for credit score checks).
- **Payment Gateway Connectors** (for EMI collection).

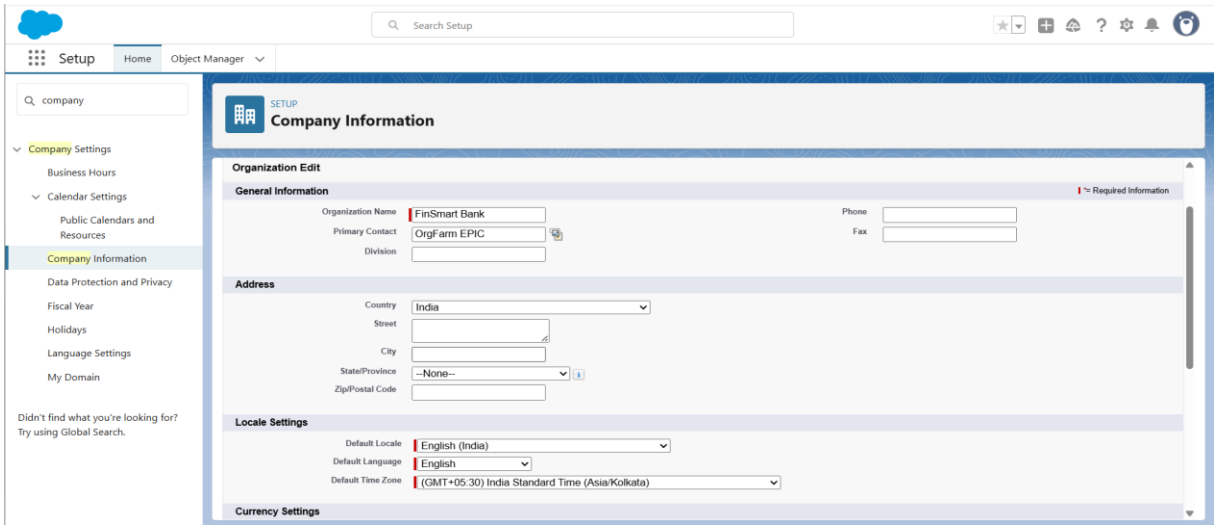
- **Document Management Apps** (for KYC storage).
- **Banking Analytics Dashboards** (pre-built loan performance report)

Phase 2: Org Setup & Configuration

1. Company Profile Setup

What I did: I configured the Company Information in Salesforce with the name 'FinSmart Bank', locale as 'English (India)', time zone as '(GMT+05:30) Kolkata', and default currency as 'INR'.

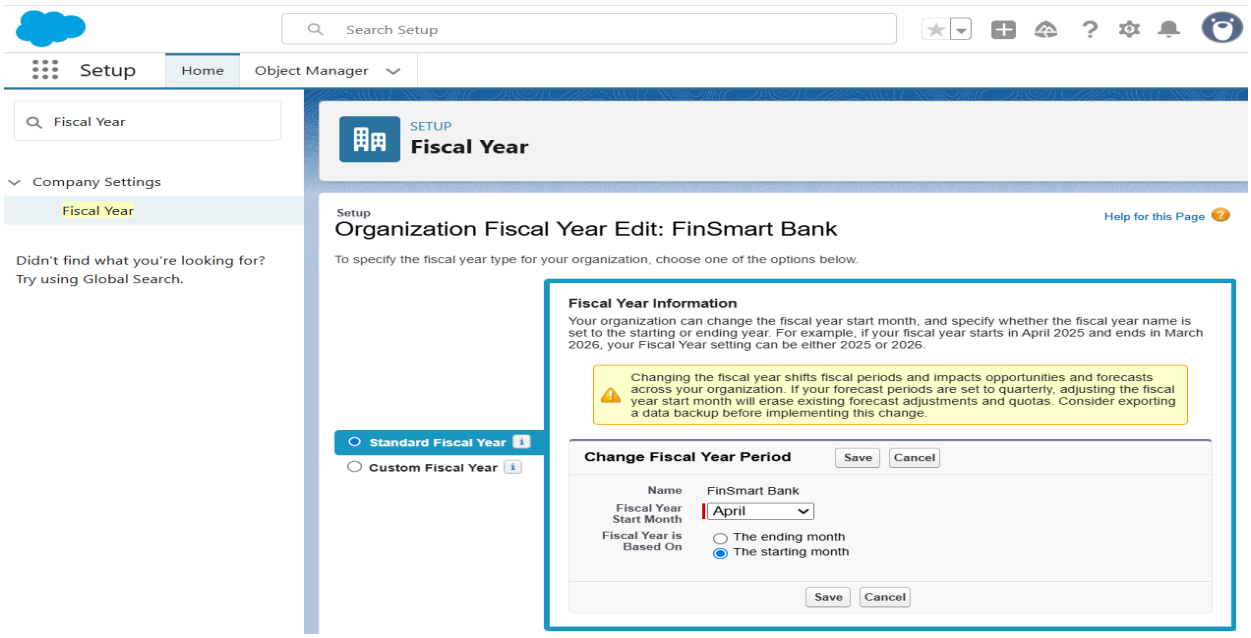
Reason: This ensures all records and transactions align with the bank's working region, and financial values are consistently reported in Indian Rupees.



2. Fiscal Year

What I did: I set up a Standard Fiscal Year starting in April.

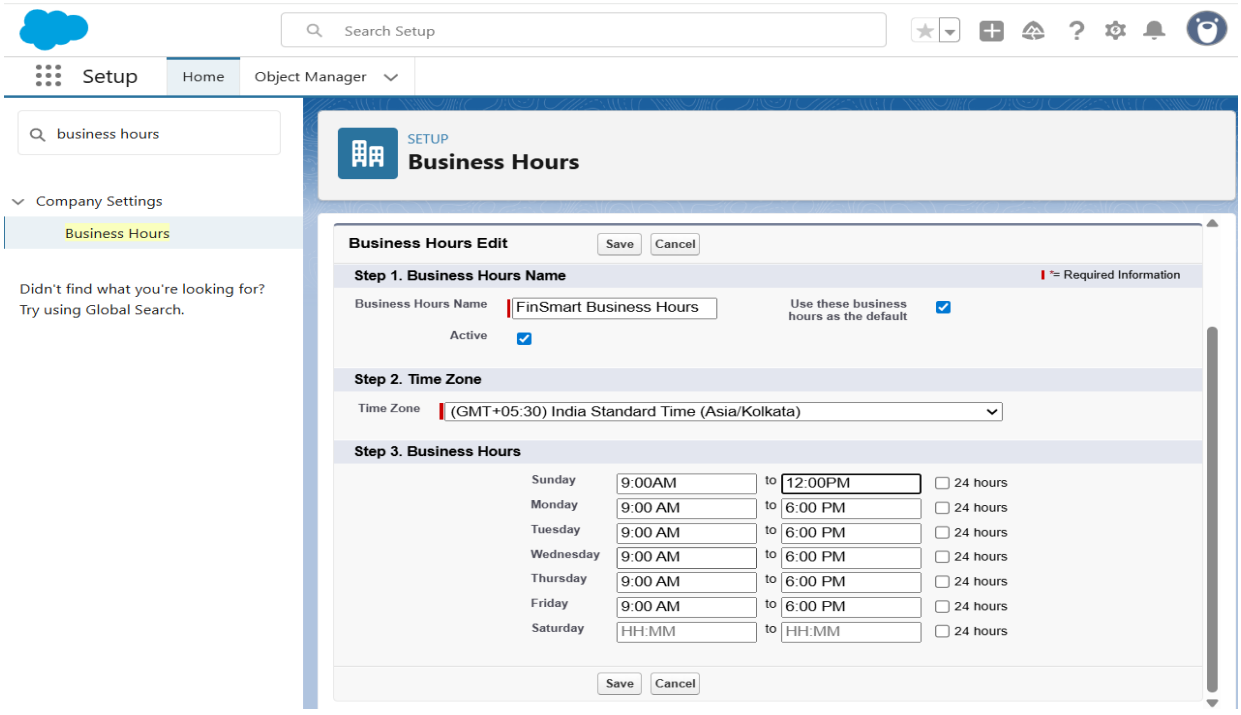
Reason: Banks in India follow the April–March financial cycle. This setup ensures reports and analytics align with regulatory and compliance needs.

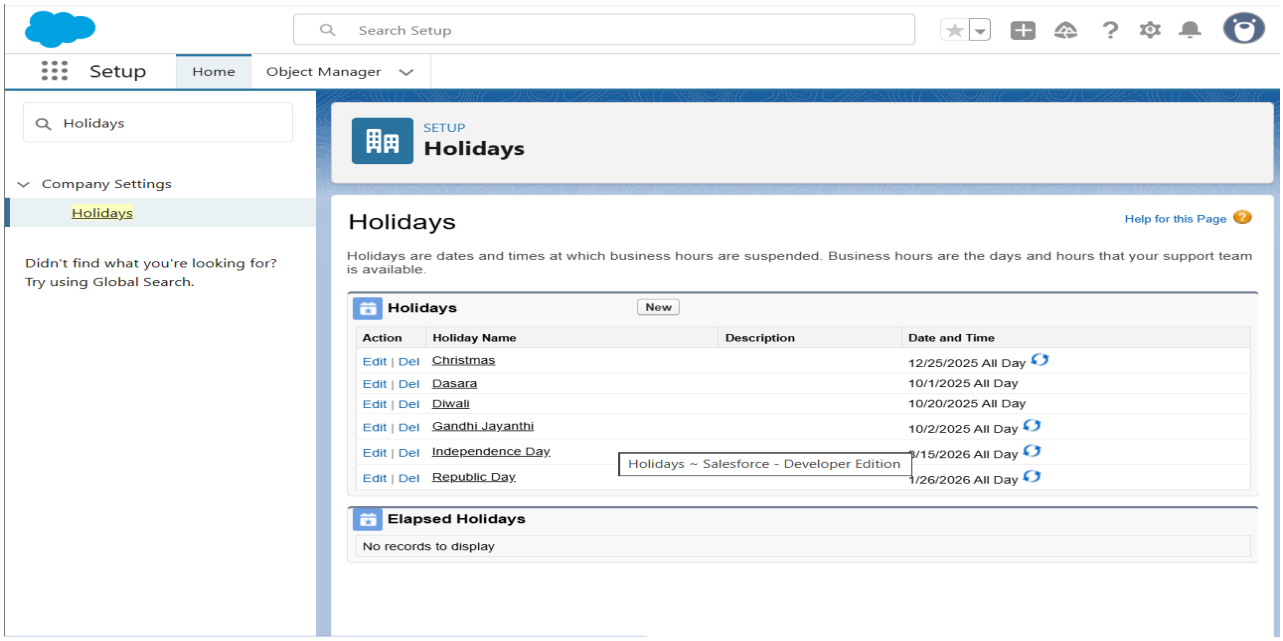


3. Business Hours & Holidays

What I did: I created business hours named 'FinSmart Business Hours' with working hours from 09:00AM to 6:00PM. I also added holidays to reflect official bank holidays.

Reason: Defining business hours ensures that service level agreements, workflows, and escalations respect actual working time. Adding holidays prevents automated tasks from running when the bank is closed.

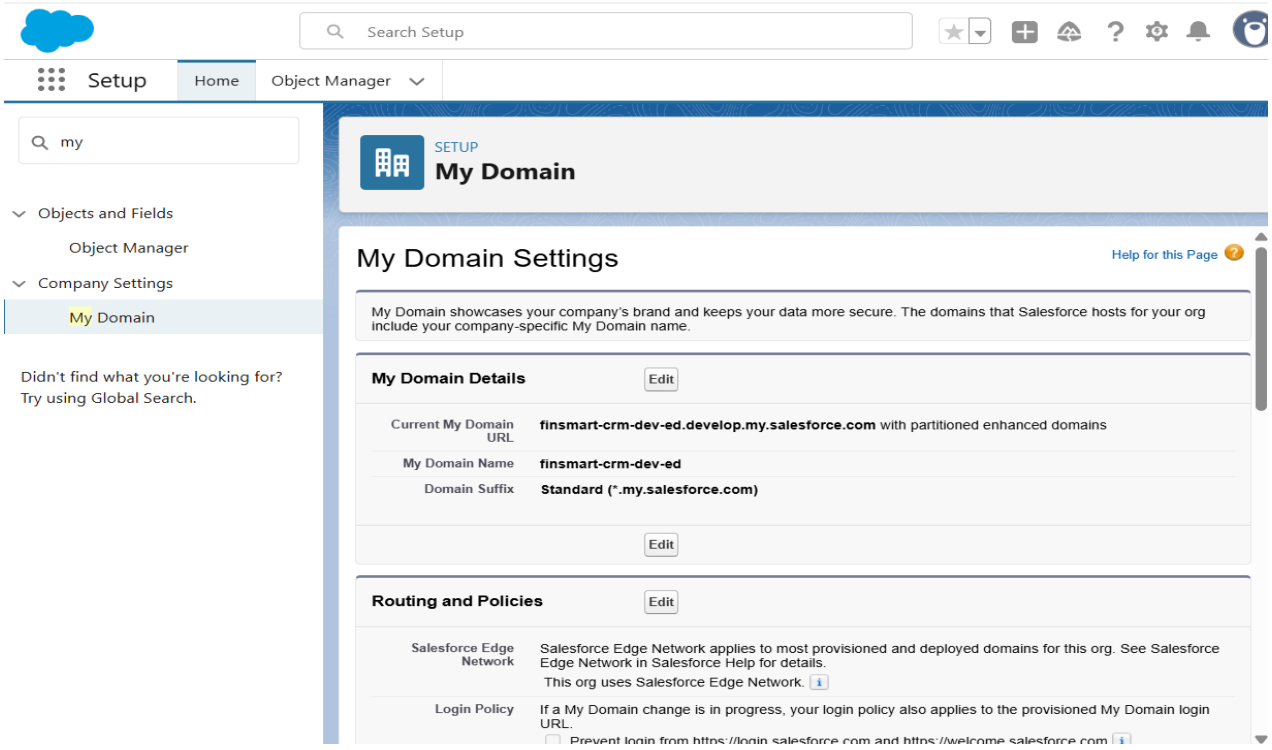




4. My Domain

What I did: I registered a unique domain name (e.g., finsmart-bank) and deployed it to users.

Reason: A My Domain is required for Lightning components, LWCs, and connected apps. It also improves branding and login security for the organization.



5. Org-Wide Defaults (OWD)

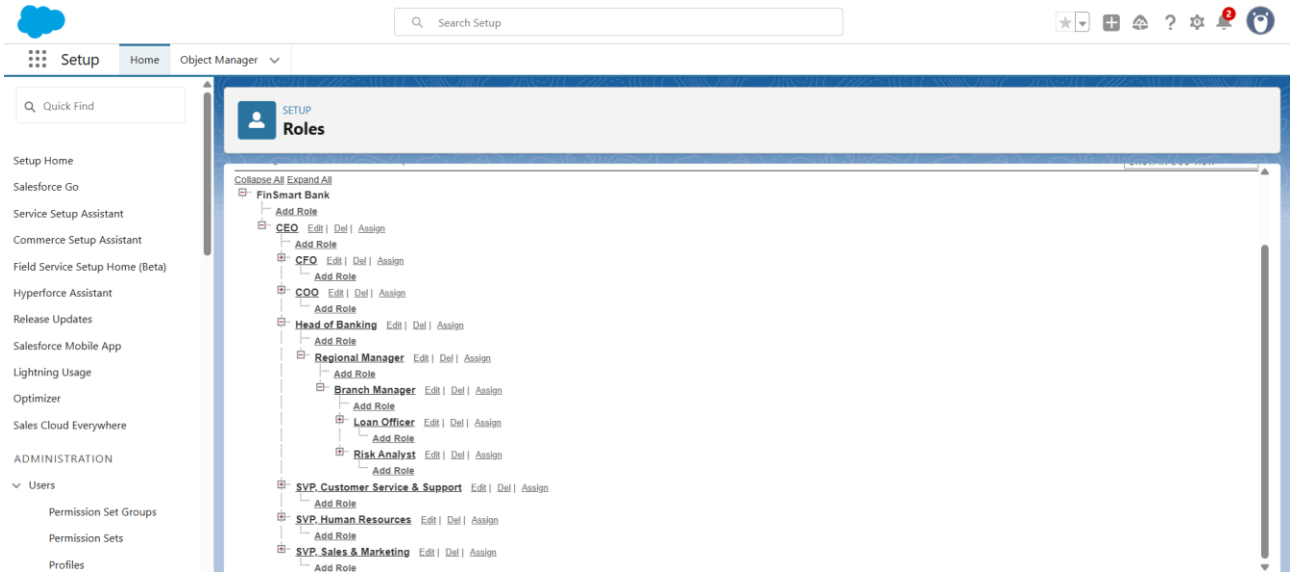
What I did: I planned OWD settings for Customer__c, Loan_Application__c, and Repayment__c as Private.

Reason: The actual configuration will be done in later stages of the project.

6. Role Hierarchy

What I did: I created a hierarchy: Head of Banking → Regional Manager → Branch Manager → (Loan Officer, Risk Analyst).

Reason: This role structure ensures proper data visibility flows upwards, allowing managers to monitor their teams while restricting unnecessary access at lower levels.

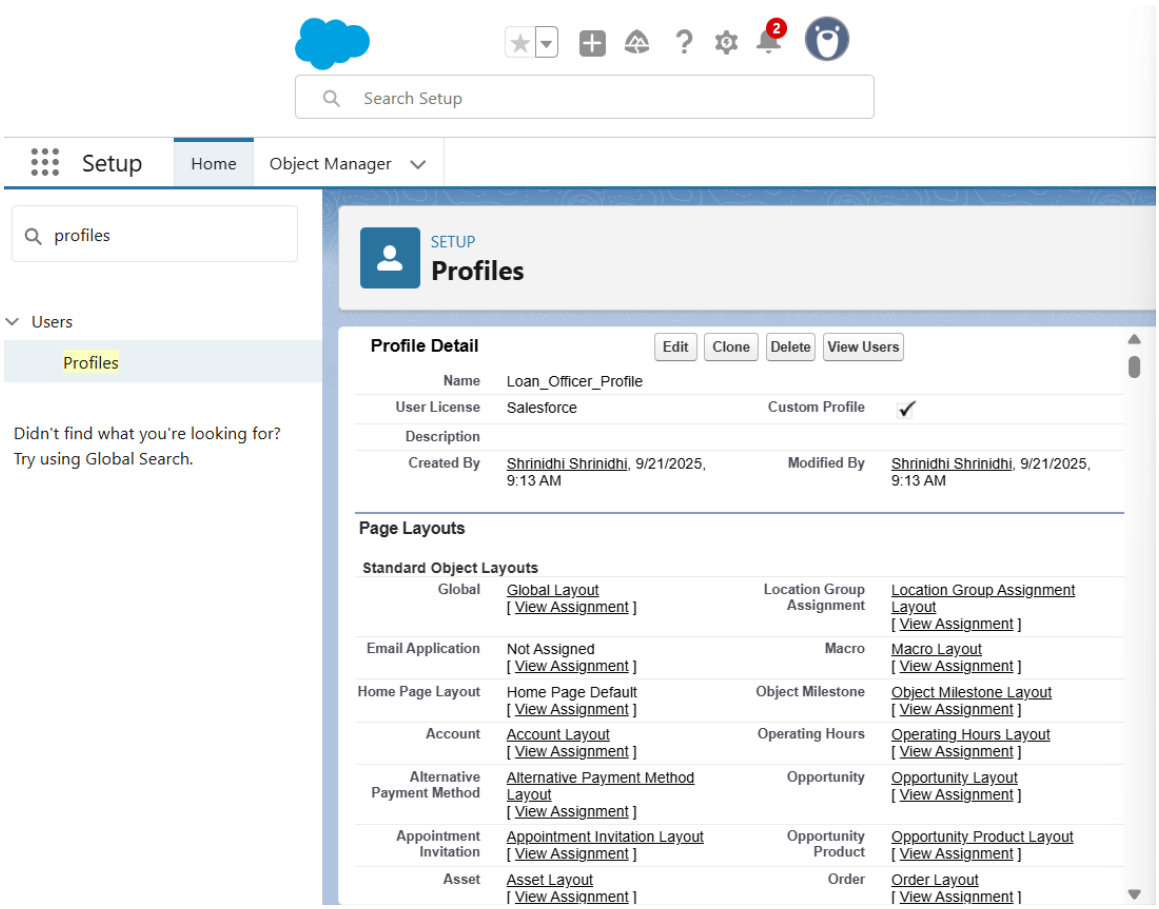


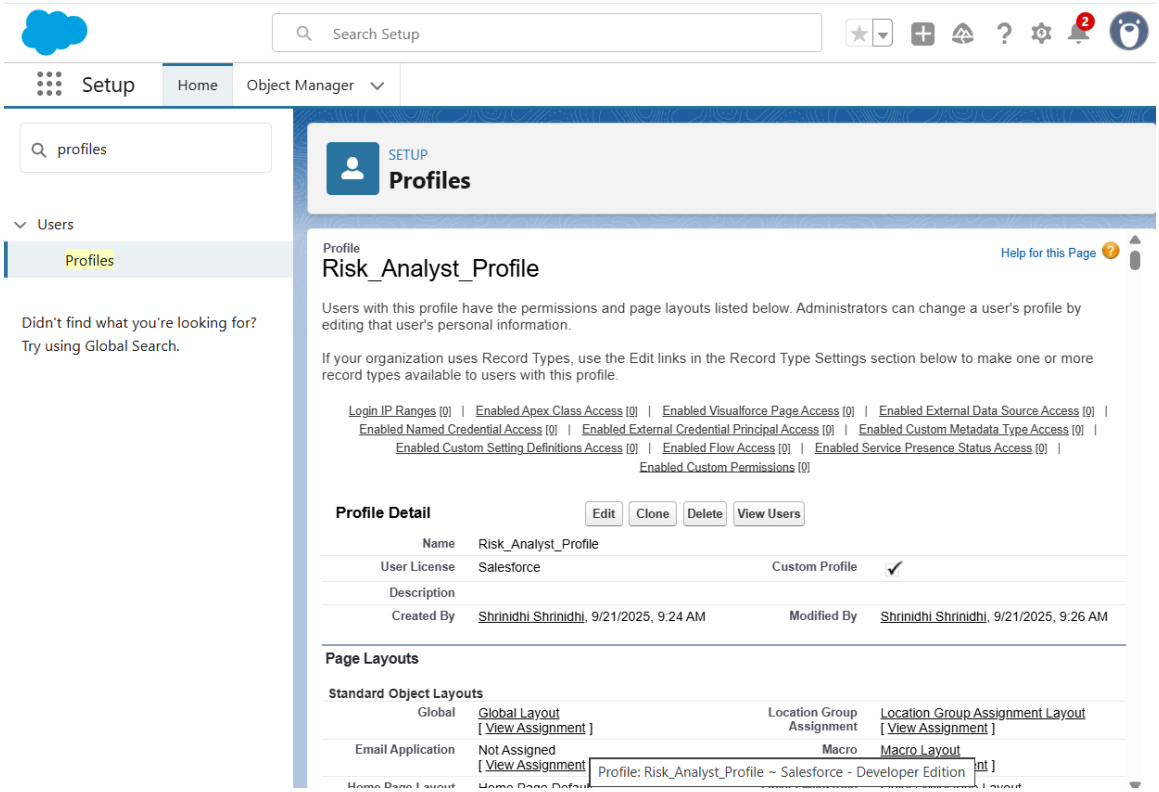
7. Profiles & Permission Sets

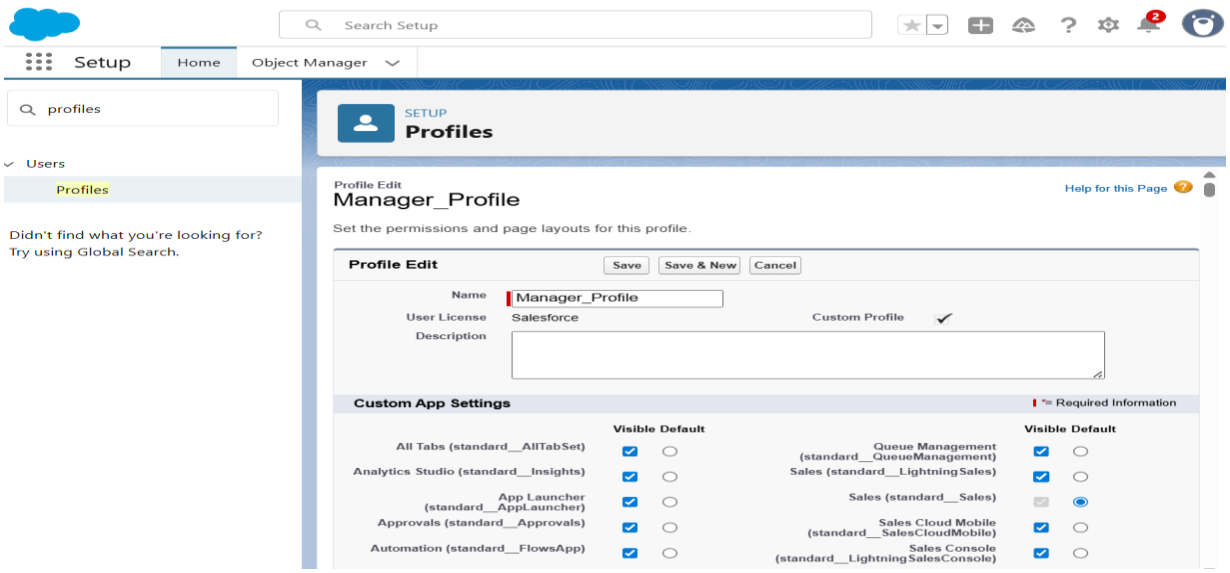
What I did: I cloned the Standard User profile to create three profiles:
- Loan_Officer_Profile – for loan officers who handle customer loan applications.
- Risk_Analyst_Profile – for risk analysts who verify documents and assess loan risk.
- Manager_Profile – for managers who oversee customers, loans, repayments, and documents.

Additionally, I planned to create a Permission Set for extra access requirements. This will be done in later phases.

Reason: Profiles control object permissions. Custom profiles were required to match responsibilities, while Permission Sets provide flexible access extensions without changing the core profiles.


Search Setup
Setup
Home
Object Manager
profiles
Users
Profiles
Didn't find what you're looking for?
Try using Global Search.
SETUP
Profiles
Profile Detail
Edit
Clone
Delete
View Users
Name
Loan_Officer_Profile
User License
Salesforce
Custom Profile
Description
Created By
Shrinidhi Shrinidhi, 9/21/2025, 9:13 AM
Modified By
Shrinidhi Shrinidhi, 9/21/2025, 9:13 AM
Page Layouts
Standard Object Layouts
Global
Global Layout
[View Assignment]
Location Group Assignment
Location Group Assignment Layout
[View Assignment]
Email Application
Not Assigned
[View Assignment]
Macro
Macro Layout
[View Assignment]
Home Page Layout
Home Page Default
[View Assignment]
Object Milestone
Object Milestone Layout
[View Assignment]
Account
Account Layout
[View Assignment]
Operating Hours
Operating Hours Layout
[View Assignment]
Alternative Payment Method
Alternative Payment Method Layout
[View Assignment]
Opportunity
Opportunity Layout
[View Assignment]
Appointment Invitation
Appointment Invitation Layout
[View Assignment]
Opportunity Product
Opportunity Product Layout
[View Assignment]
Asset
Asset Layout
[View Assignment]
Order
Order Layout
[View Assignment]


Search Setup
Setup
Home
Object Manager
profiles
Users
Profiles
Didn't find what you're looking for?
Try using Global Search.
SETUP
Profiles
Profile
Risk_Analyst_Profile
Help for this Page
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.
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.
Login IP Ranges [0] | Enabled Apex Class Access [0] | Enabled Visualforce Page Access [0] | Enabled External Data Source Access [0] | Enabled Named Credential Access [0] | Enabled External Credential Principal Access [0] | Enabled Custom Metadata Type Access [0] | Enabled Custom Setting Definitions Access [0] | Enabled Flow Access [0] | Enabled Service Presence Status Access [0] | Enabled Custom Permissions [0]
Profile Detail
Edit
Clone
Delete
View Users
Name
Risk_Analyst_Profile
User License
Salesforce
Custom Profile
Description
Created By
Shrinidhi Shrinidhi, 9/21/2025, 9:24 AM
Modified By
Shrinidhi Shrinidhi, 9/21/2025, 9:26 AM
Page Layouts
Standard Object Layouts
Global
Global Layout
[View Assignment]
Location Group Assignment
Location Group Assignment Layout
[View Assignment]
Email Application
Not Assigned
[View Assignment]
Macro
Macro Layout
Profile: Risk_Analyst_Profile ~ Salesforce - Developer Edition

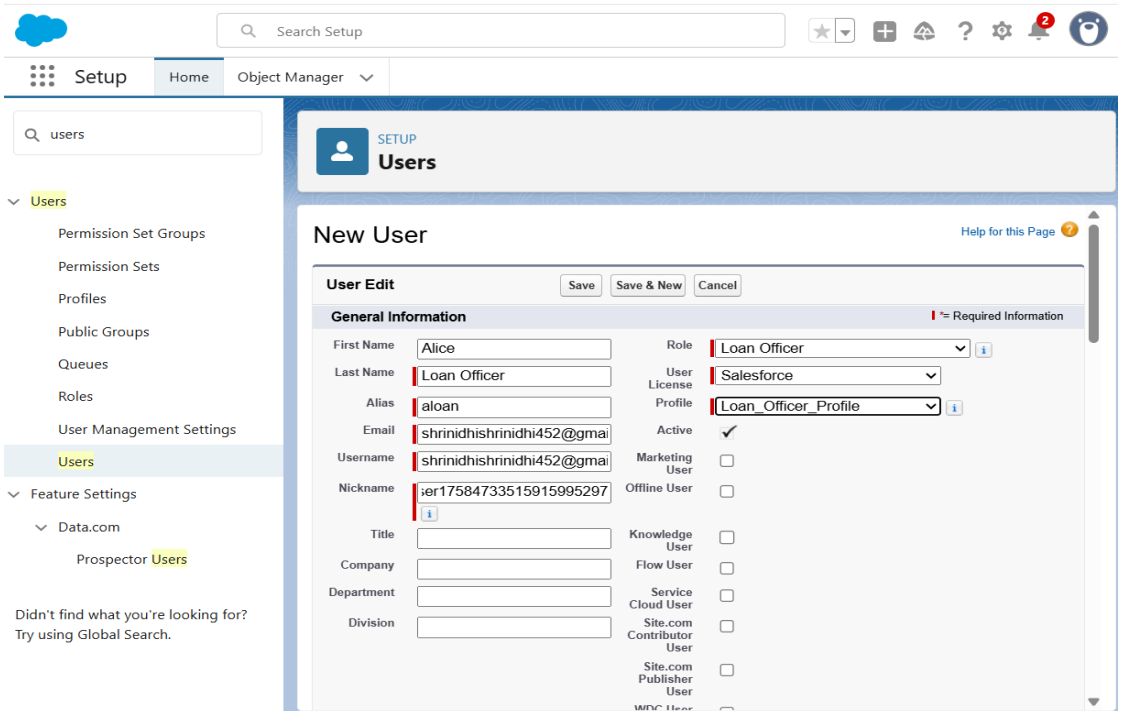


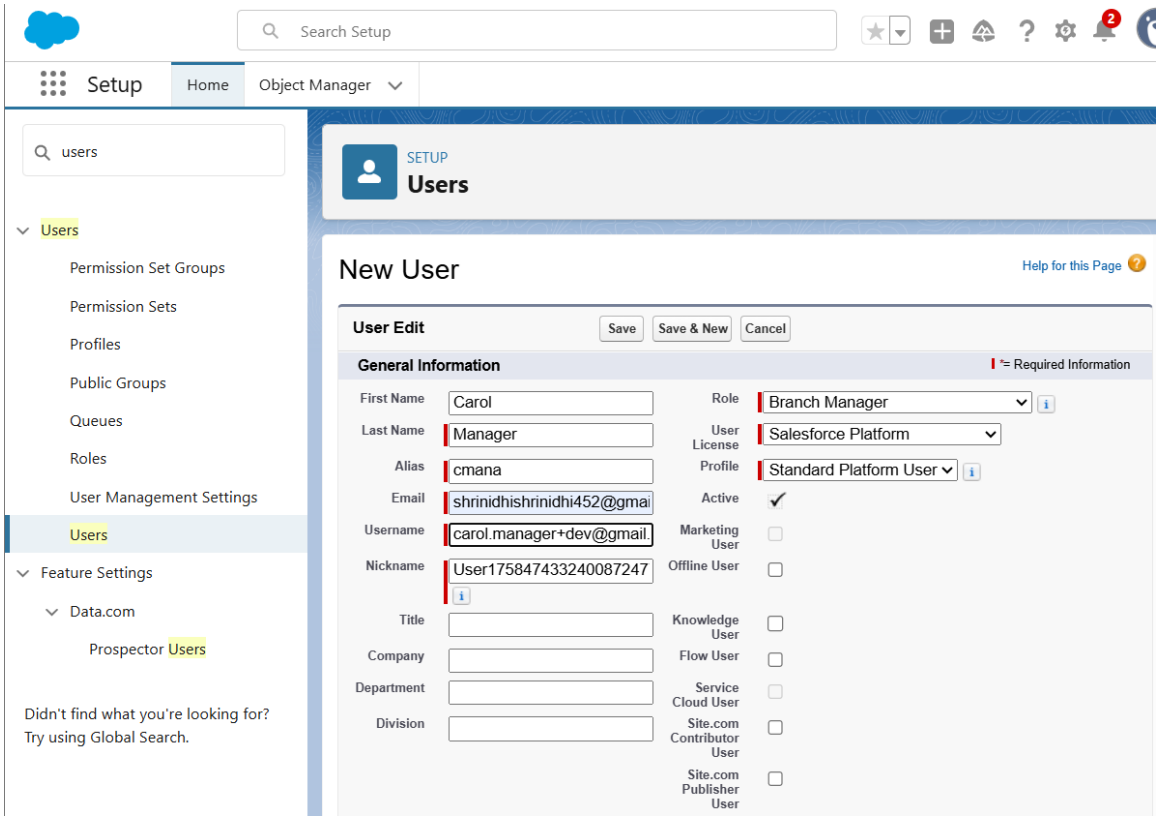
8. Users & Licenses

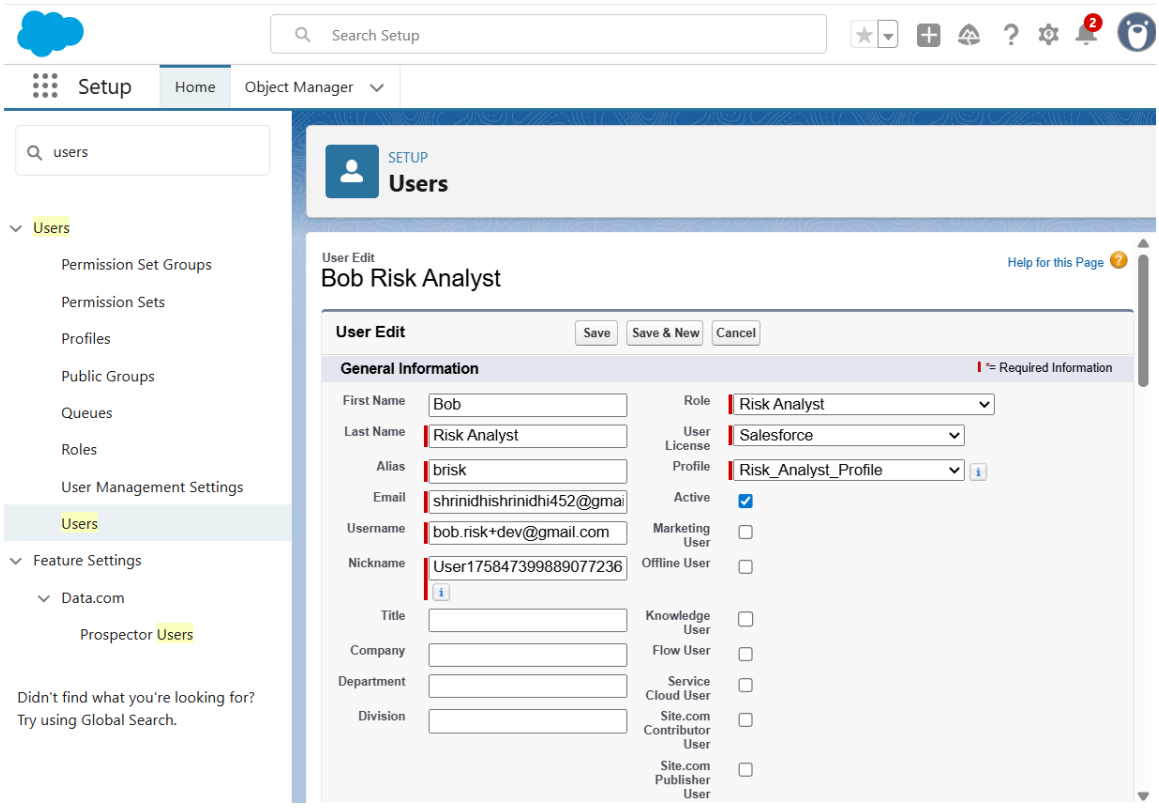
What I did: I created three test users:

- Alice – Loan Officer with Loan_Officer_Profile.
- Bob – Risk Analyst with Risk_Analyst_Profile.
- Carol – Manager with Manager_Profile.

Reason: These test users simulate different roles in the bank. They allow testing of profile permissions, role hierarchy, and data visibility to ensure proper security implementation.




Search Setup
Setup
Home
Object Manager
users
Users
Permission Set Groups
Permission Sets
Profiles
Public Groups
Queues
Roles
User Management Settings
Users
Feature Settings
Data.com
Prospector Users
Didn't find what you're looking for?
Try using Global Search.
SETUP
Users
New User
Help for this Page
User Edit
Save
Save & New
Cancel
General Information
= Required Information
First Name
Carol
Last Name
Manager
Alias
cmana
Email
shrinidhishrinidhi452@gmai
Username
carol.manager+dev@gmail.
Nickname
User17584733240087247
Title
Company
Department
Division
Role
Branch Manager
User License
Salesforce Platform
Profile
Standard Platform User
Active
Marketing User
Offline User
Knowledge User
Flow User
Service Cloud User
Site.com Contributor User
Site.com Publisher User


Search Setup
Setup
Home
Object Manager
users
Users
Permission Set Groups
Permission Sets
Profiles
Public Groups
Queues
Roles
User Management Settings
Users
Feature Settings
Data.com
Prospector Users
Didn't find what you're looking for?
Try using Global Search.
SETUP
Users
User Edit
Bob Risk Analyst
Help for this Page
User Edit
Save
Save & New
Cancel
General Information
= Required Information
First Name
Bob
Last Name
Risk Analyst
Alias
brisk
Email
shrinidhishrinidhi452@gmai
Username
bob.risk+dev@gmail.com
Nickname
User17584739988907723 6
Title
Company
Department
Division
Role
Risk Analyst
User License
Salesforce
Profile
Risk_Analyst_Profile
Active
Marketing User
Offline User
Knowledge User
Flow User
Service Cloud User
Site.com Contributor User
Site.com Publisher User

9. Field History Tracking

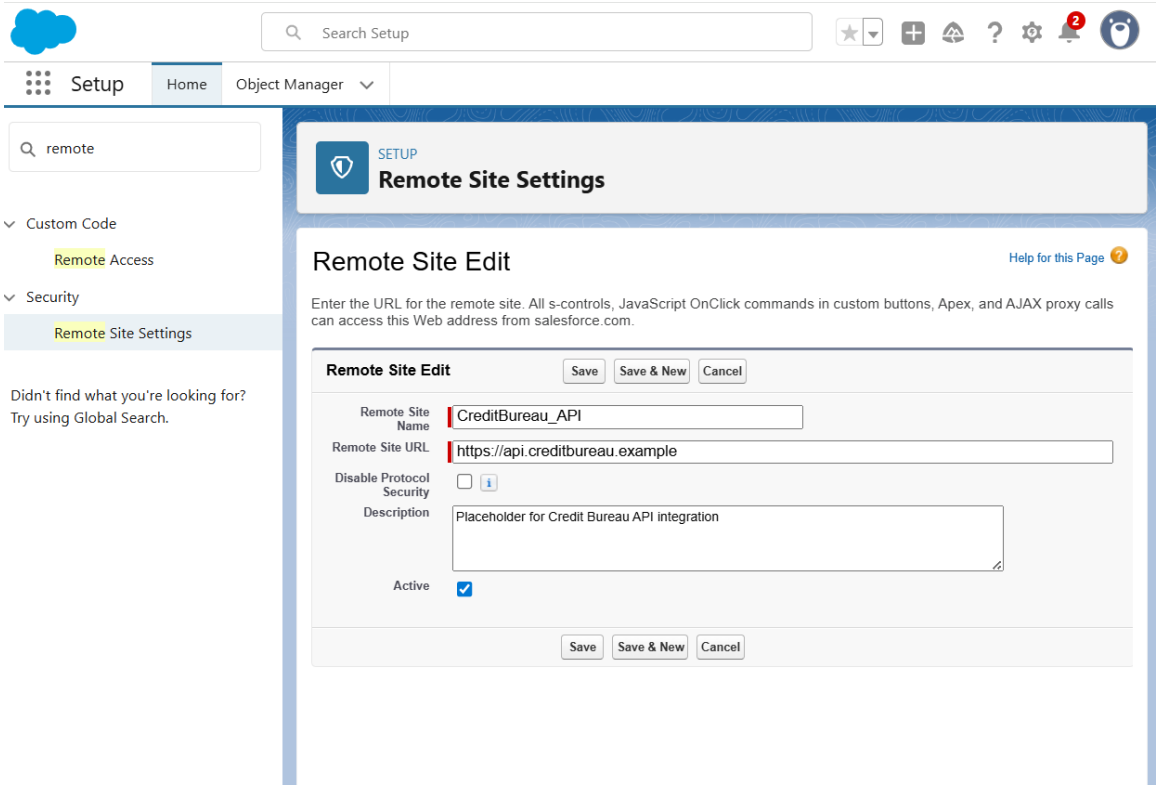
What I did: I enabled history tracking for key fields: Loan_Status__c, Loan_Amount__c, Assigned_To__c (Loan Applications); Payment_Status__c, Paid_Amount__c (Repayments); Verification_Status__c, Expiry_Date__c (Documents); Offer_Status__c, Interest_Rate__c (Offers).

Reason: This ensures an audit trail of critical financial and customer-related fields, supporting compliance, accountability, and transparency.

10. Remote Site Settings & Named Credentials

What I did: I added a Remote Site setting for CreditBureau_API and created a Named Credential (NamedCredential_CreditBureau).

Reason: Remote Site Settings and Named Credentials are required for secure integrations with external systems like credit bureau APIs, allowing automated credit checks during loan processing.



Phase 3: Data Modeling &Relationships

In this phase, I created custom objects, fields, record types, page layouts, and relationships to model the FinSmart Bank CRM system. Screenshots are attached for each major step.

1. Custom Objects Creation

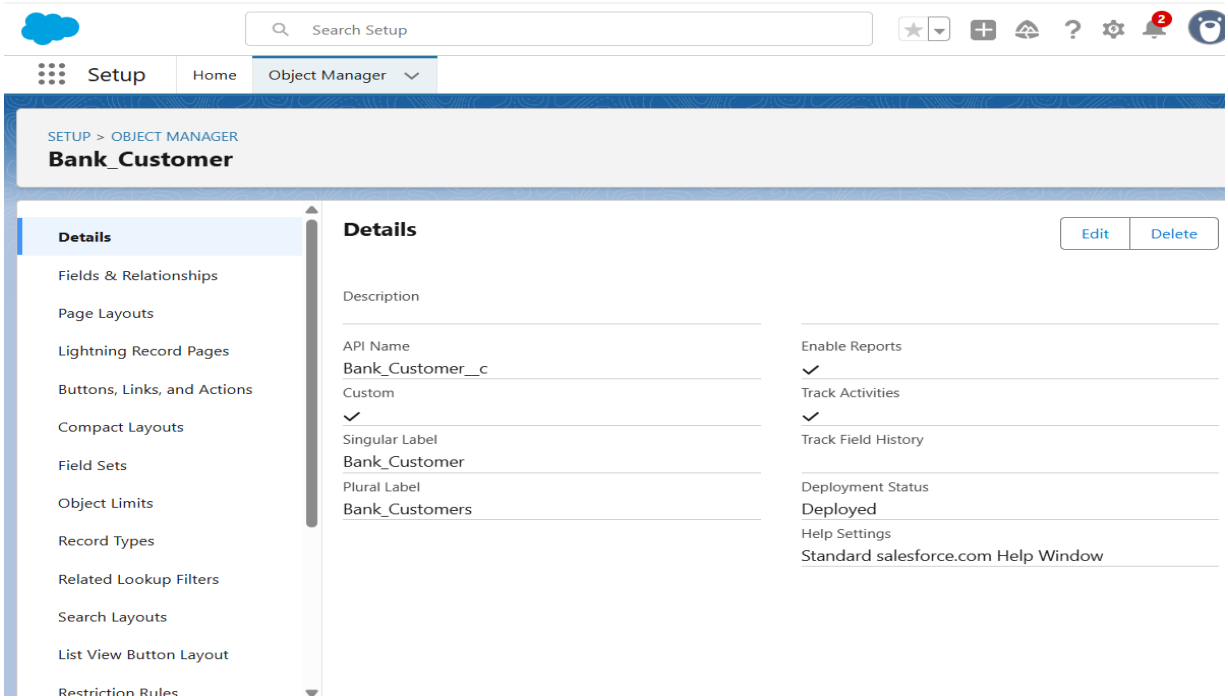
What I did: I created the following custom objects:

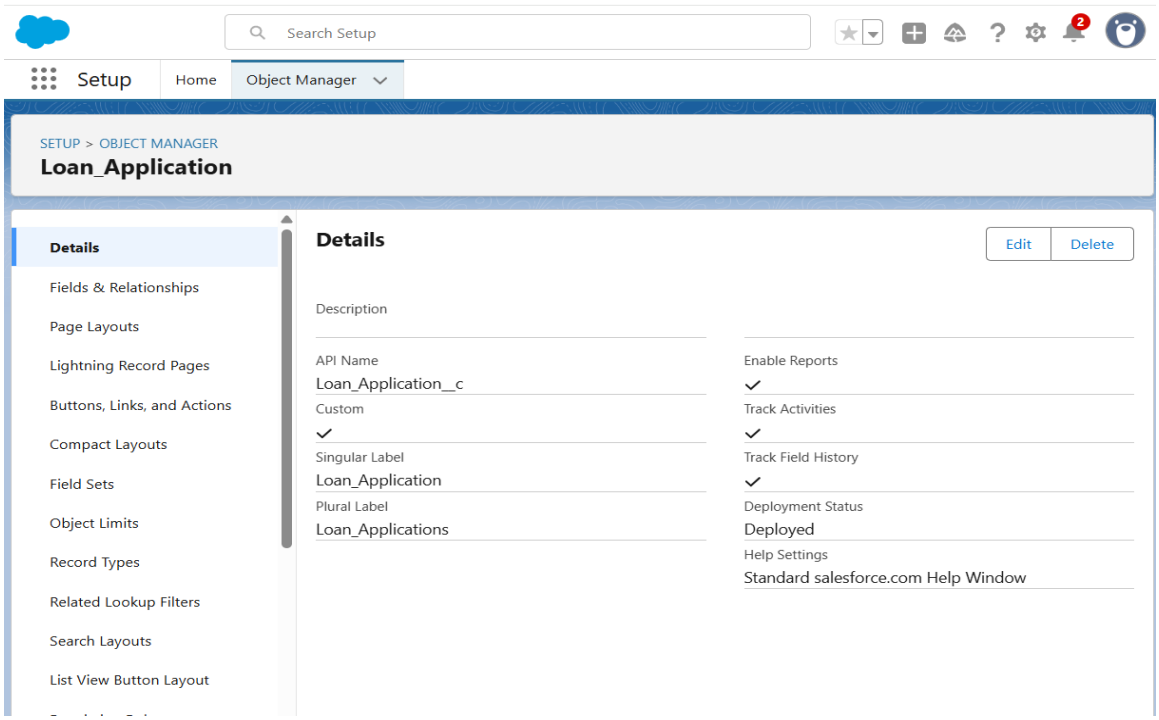
- Bank_Customer__c (Label: Bank_Customer)
- Loan_Application__c (Label: Loan _Application)
- Repayment__c (Label: Repayment)
- Document__c (Label: Document)
- Offer__c (Label: Offer)

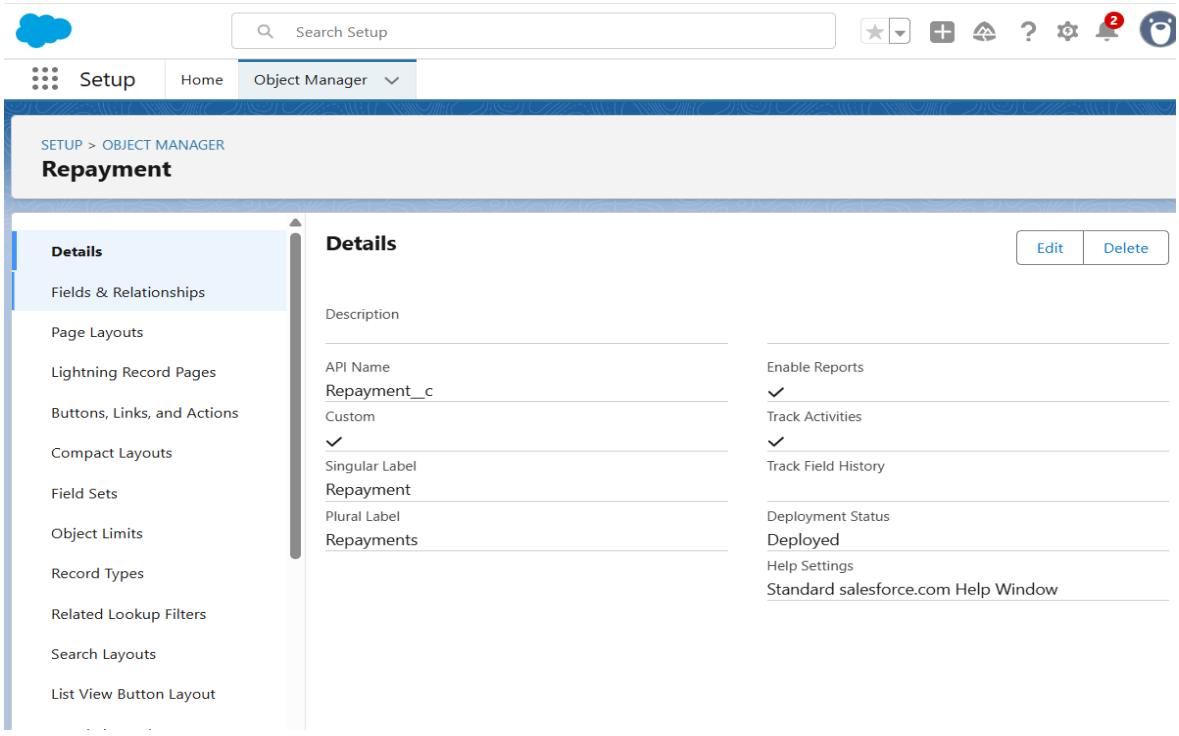
For Loan_Application__c and Repayment__c, I used Auto-Number record names to ensure unique identifiers (e.g., LA-{0000}, RP-{0000}). For Bank_Customer__c, I used a text record name to capture full customer names.

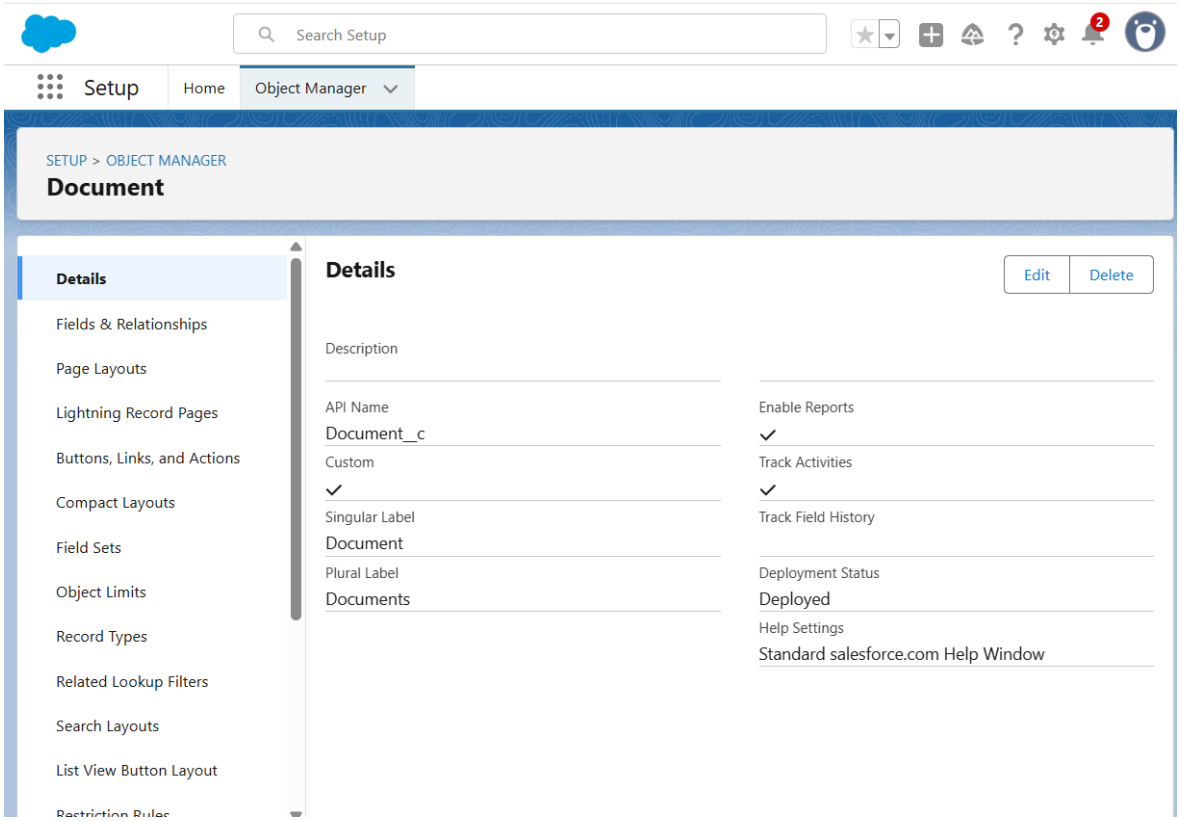
- I created custom **tabs** for Bank_Customer, Loan_Application, Repayment, Document, and Offer so that users can easily access these objects from the Salesforce app navigation bar.

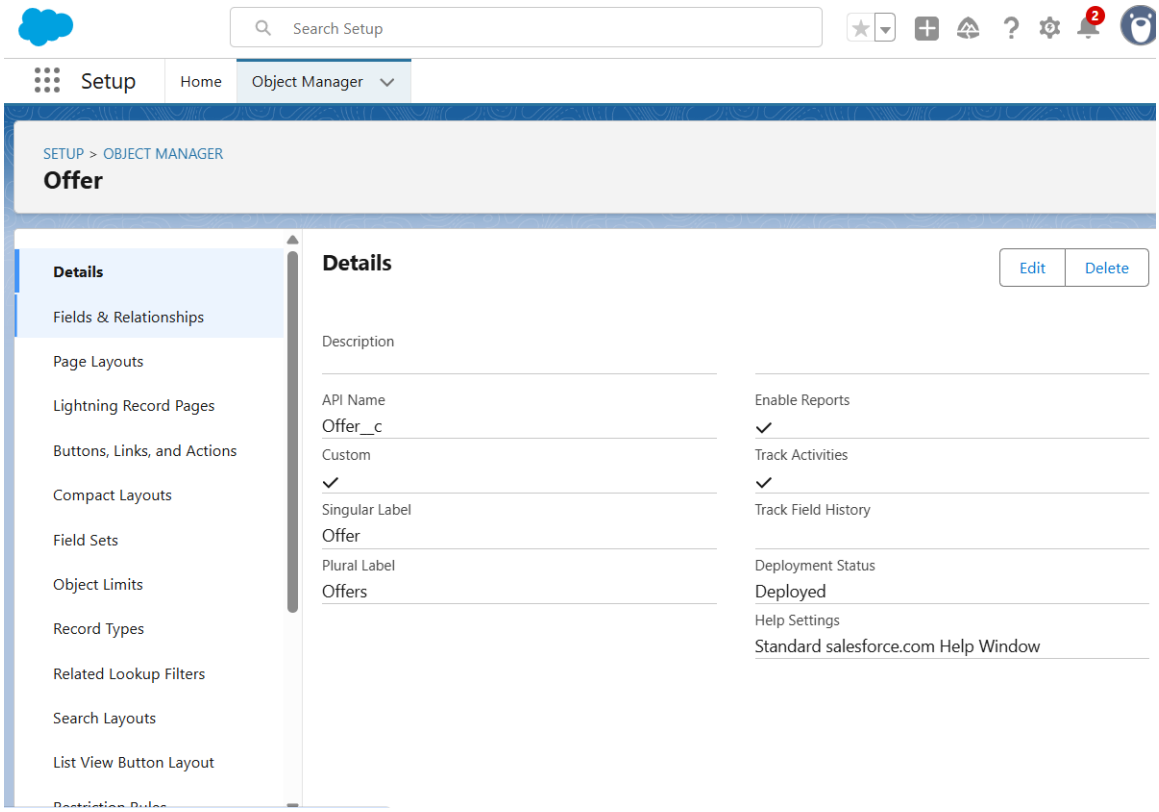
Reason: These objects represent the core banking entities needed for loan processing, repayments, document tracking, and offers. Using separate objects ensures proper data organization and relationships.




Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Loan_Application
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Details
Edit
Delete
Description
API Name
Loan_Application__c
Custom
Singular Label
Loan_Application
Plural Label
Loan_Applications
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Repayment
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Details
Edit
Delete
Description
API Name
Repayment__c
Custom
Singular Label
Repayment
Plural Label
Repayments
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window

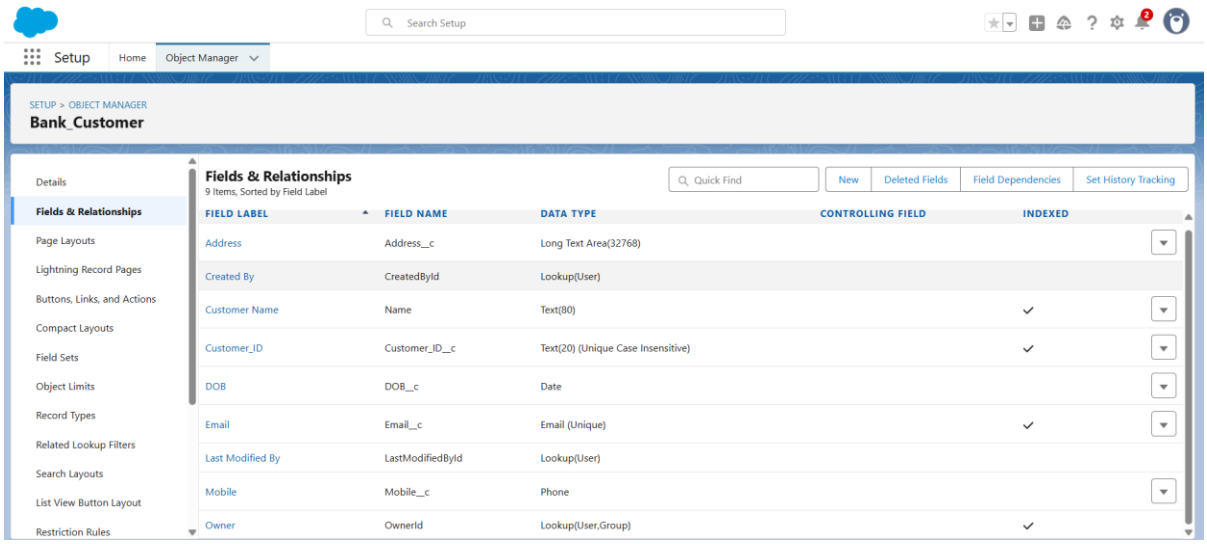

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Document
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Details
Edit
Delete
Description
API Name
Document__c
Custom
Singular Label
Document
Plural Label
Documents
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Offer
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Details
Edit
Delete
Description
API Name
Offer__c
Custom
Singular Label
Offer
Plural Label
Offers
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window

2. Key Fields for Bank_Customer__c

What I did: I created key fields on Bank_Customer__c:
• Email__c (Email)
• Mobile__c (Phone)
• Customer_ID__c (Text, 20)
• DOB__c (Date)
• Address__c (Long Text Area)

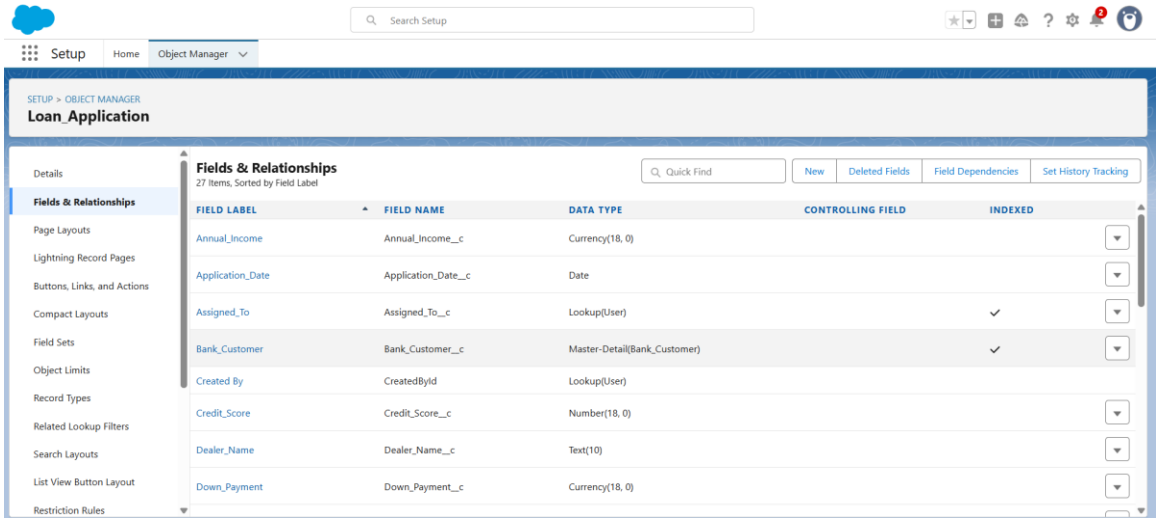
Reason: These fields capture essential customer information required for KYC compliance and loan processing.



3. Key Fields for Loan_Application__c

What I did: I added key fields on Loan_Application__c:
• Customer__c (Master-Detail to Bank_Customer__c)
• Loan_Type__c (Picklist: Home, Car, Personal, Education, Gold)
• Loan_Amount__c (Currency)
• Interest_Rate__c (Percent)
• Tenure_Months__c (Number)
• EMI__c (Currency, calculated later by Flow/Apex)
• Loan_Status__c (Picklist: Draft, Submitted, Under_Review, Approved, Rejected, Disbursed, Active, Closed, Defaulted)
• Credit_Score__c (Number)
• Assigned_To__c (Lookup to User)
• Application_Date__c, Disbursement_Date__c (Date fields)

Reason: These fields track all key attributes of a loan application, including type, status, amount, repayment terms, and ownership.

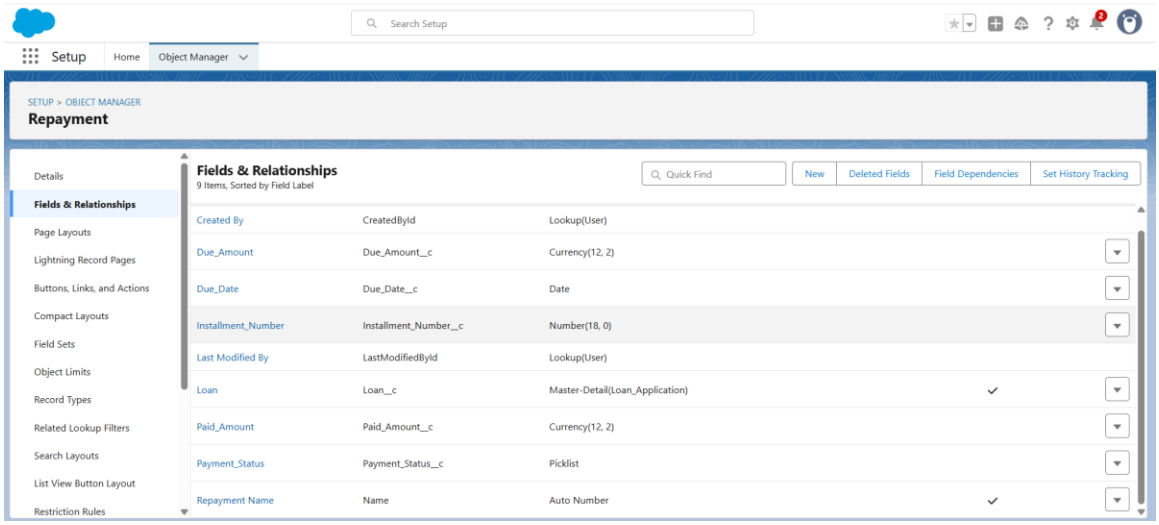


4. Key Fields for Repayment__c

What I did: I created the following fields on Repayment__c:

- Loan__c (Master-Detail to Loan_Application__c)
- Installment_Number__c (Number)
- Due_Date__c, Payment_Date__c (Date)
- Due_Amount__c, Paid_Amount__c (Currency)
- Payment_Status__c (Picklist: Due, Paid, Overdue, Partially_Paid)

Reason: This object allows tracking of repayment schedules, ensuring each installment is monitored accurately.

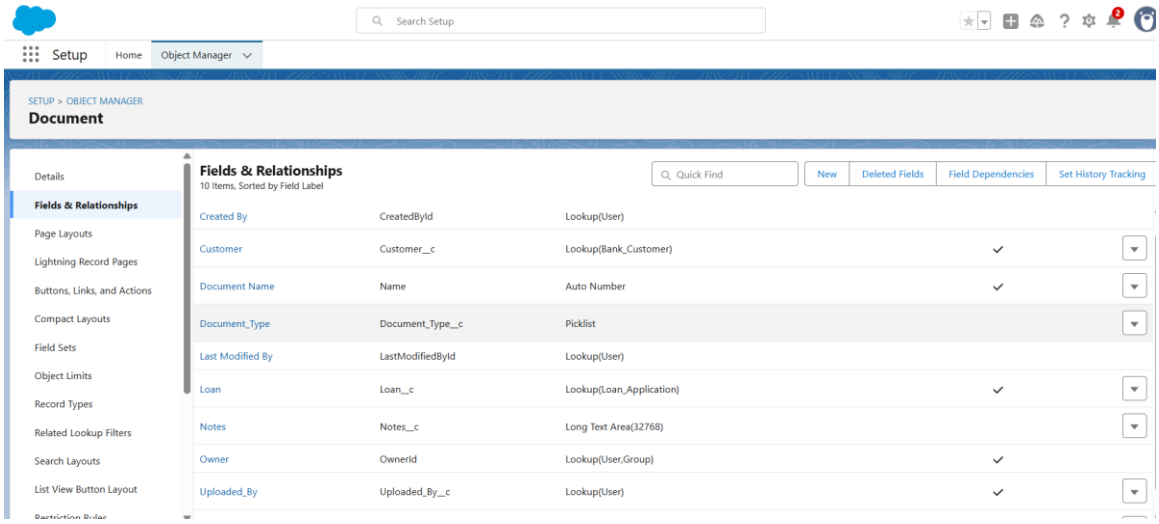


5. Key Fields for Document__c

What I did: I added the following fields:

• Related_Loan__c (Lookup to Loan_Application__c)
• Bank_Customer__c (Optional Lookup)
• Document_Type__c (Picklist: PAN, Aadhaar, Salary Slip, Bank Statement, Address Proof, Other)
• Document_Number__c (Text)
• Issue_Date__c, Expiry_Date__c (Date)
• Verification_Status__c (Picklist: Pending, Verified, Rejected)
• Uploaded_By__c (Lookup to User)
• Notes__c (Long Text Area)

Reason: Document__c stores metadata for KYC and proof documents, while the actual files are managed using Salesforce Files.

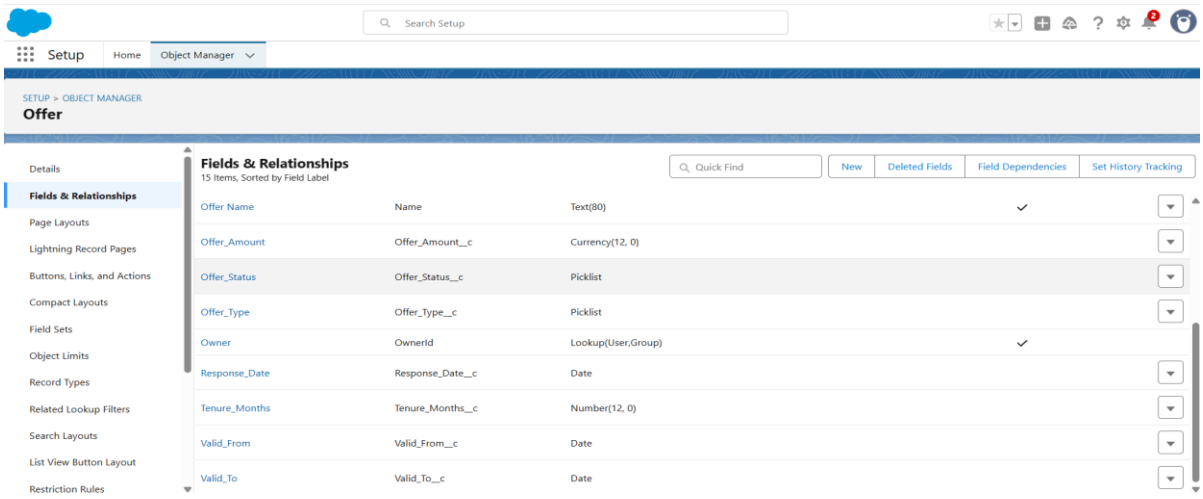


6. Key Fields for Offer__c

What I did: I created fields for Offer__c:

• Bank_Customer__c (Lookup)
• Offer_Type__c (Picklist: Personal Loan, Home Loan, Car Loan, Credit Card, Insurance)
• Offer_Amount__c (Currency)
• Interest_Rate__c (Percent)
• Tenure_Months__c (Number)
• Offer_Status__c (Picklist: Draft, Sent, Accepted, Rejected, Expired)
• Valid_From__c, Valid_To__c (Date)
• Generated_By__c (Lookup User)
• Response_Date__c (Date)
• Notes__c (Long Text Area)

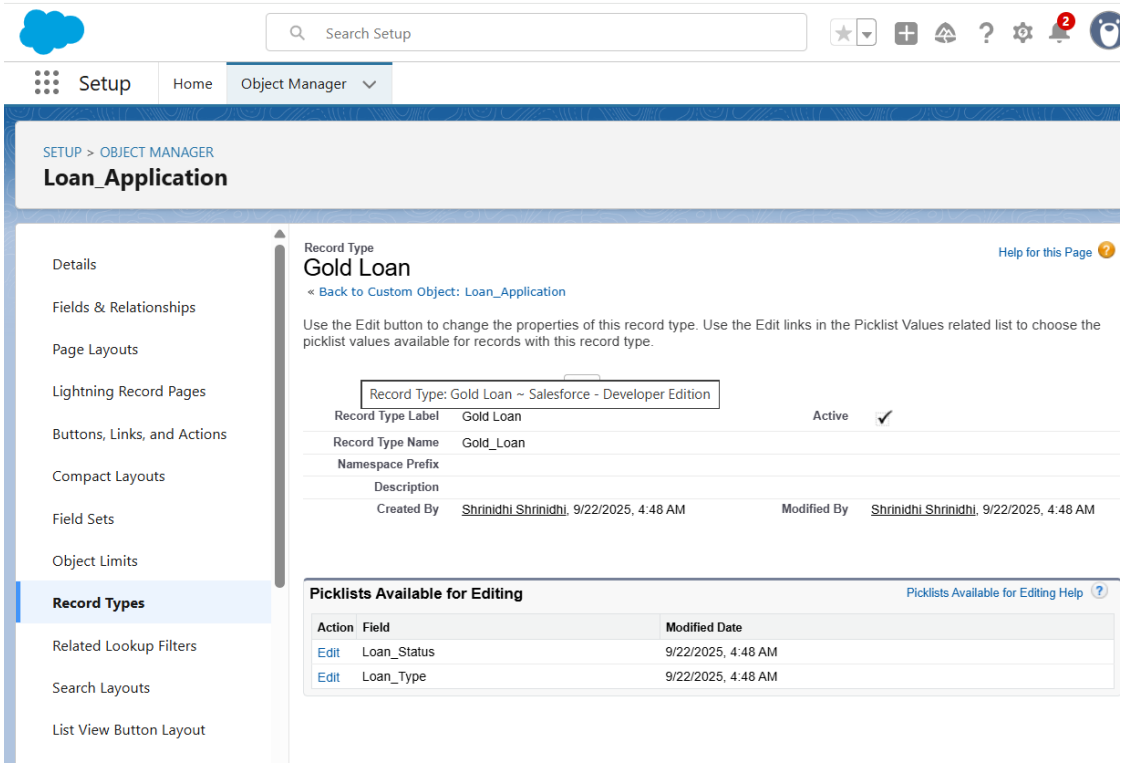
Reason: Offer__c manages pre-approved or promotional loan offers, which can be accepted by customers to generate loan applications automatically.

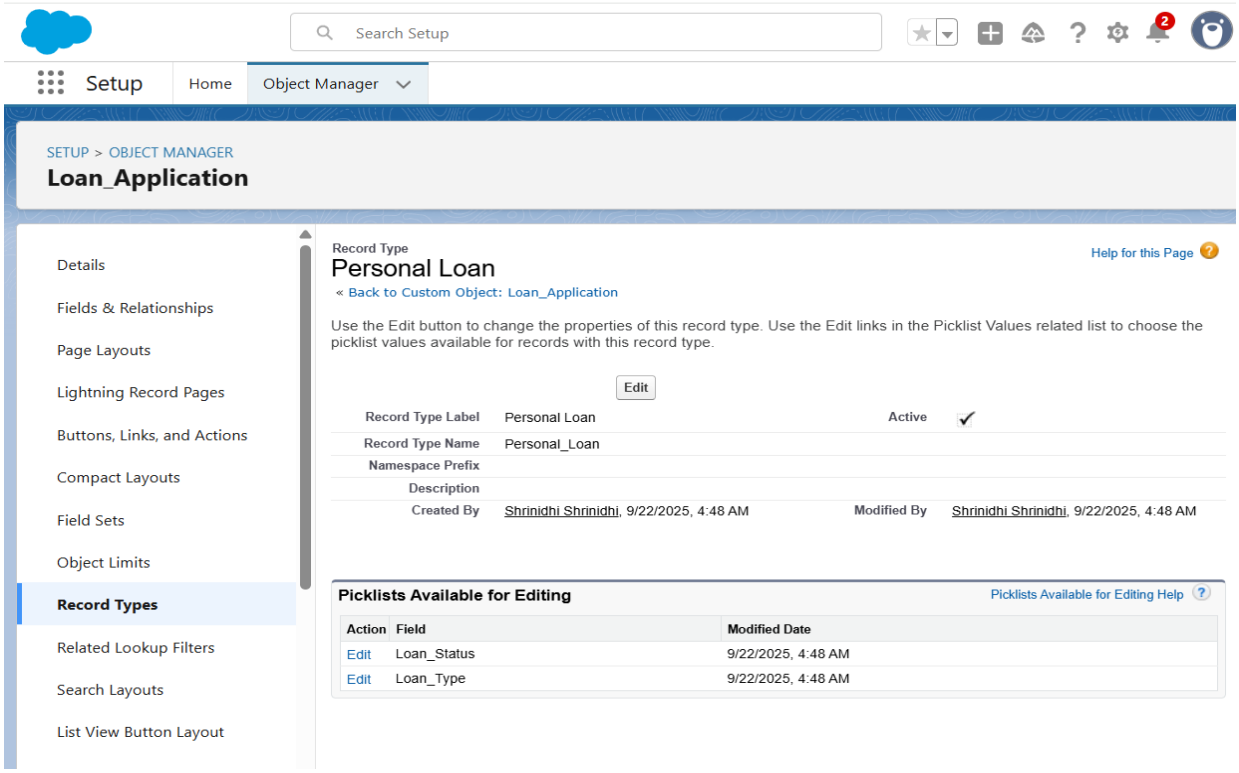


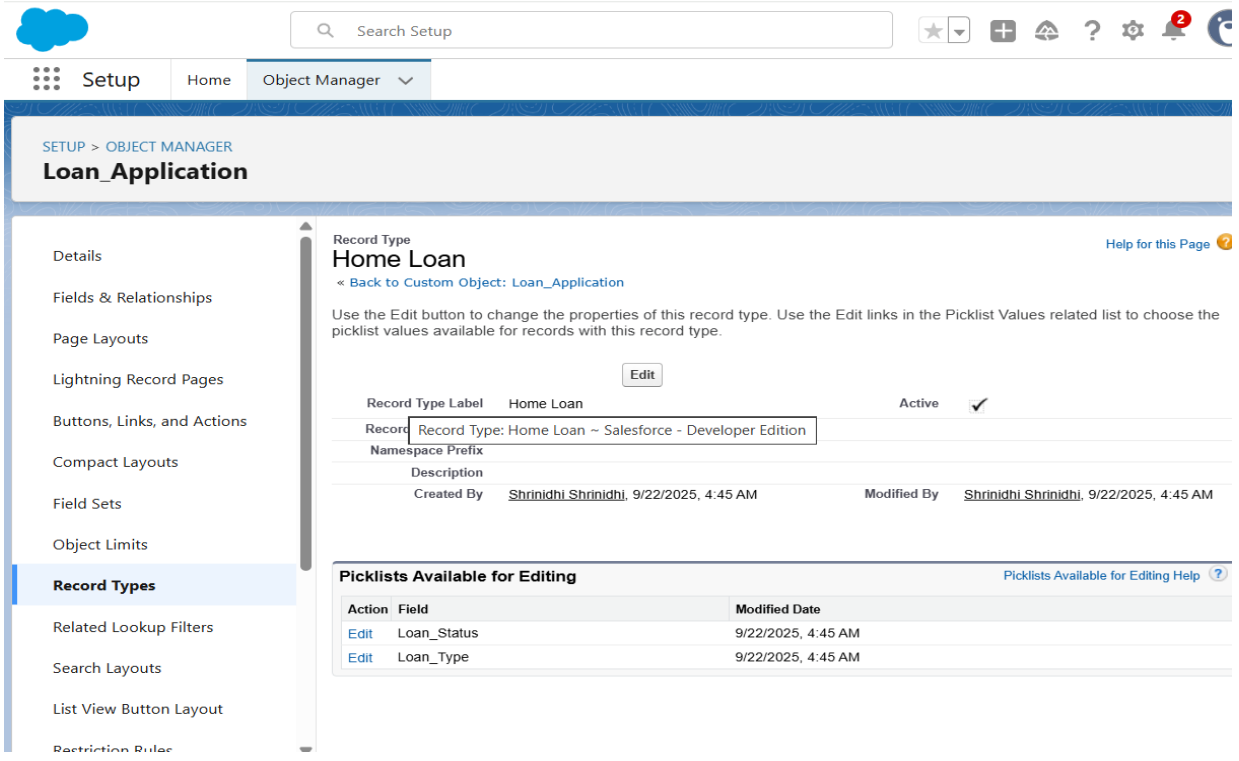
7. Record Types & Page Layouts

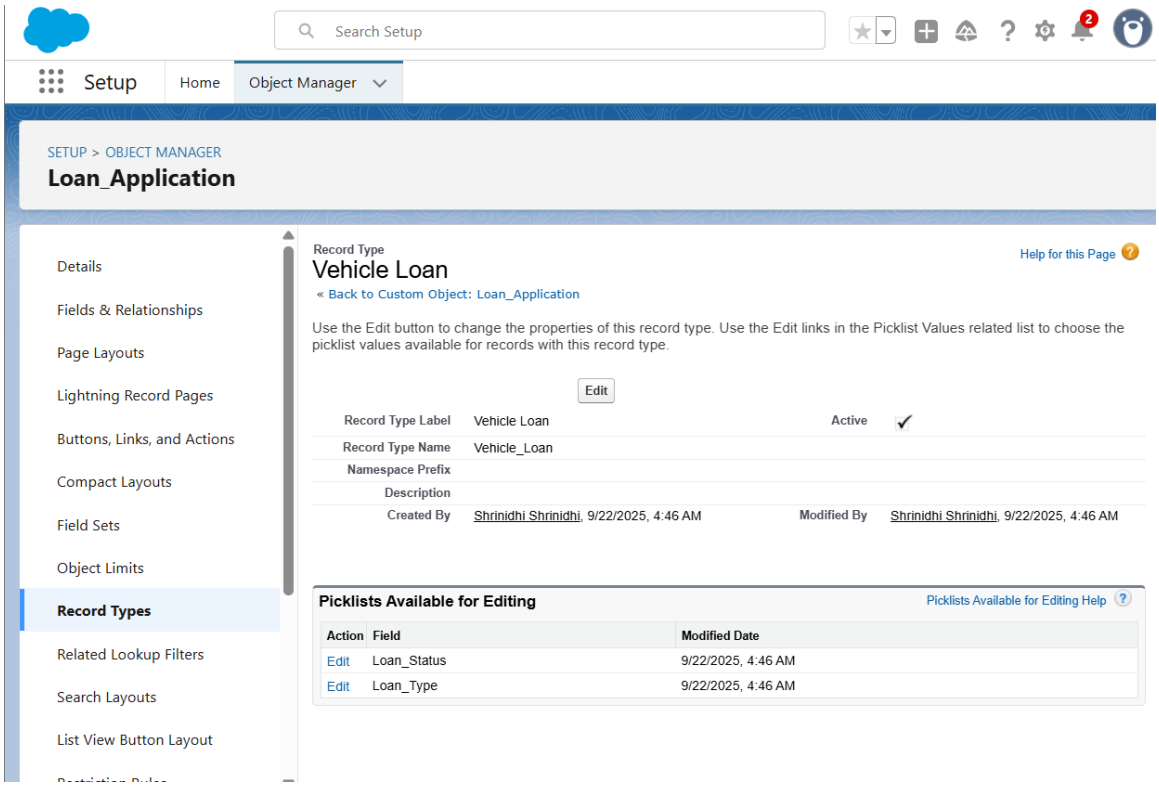
What I did: For Loan_Application__c, I created Record Types for Home Loan, Vehicle Loan, Personal Loan, and Gold Loan.
Each record type has its own page layout customized with fields relevant to that loan type.

Reason: Record Types allow tailoring loan application forms to specific loan categories, making data entry simpler and more relevant for users.

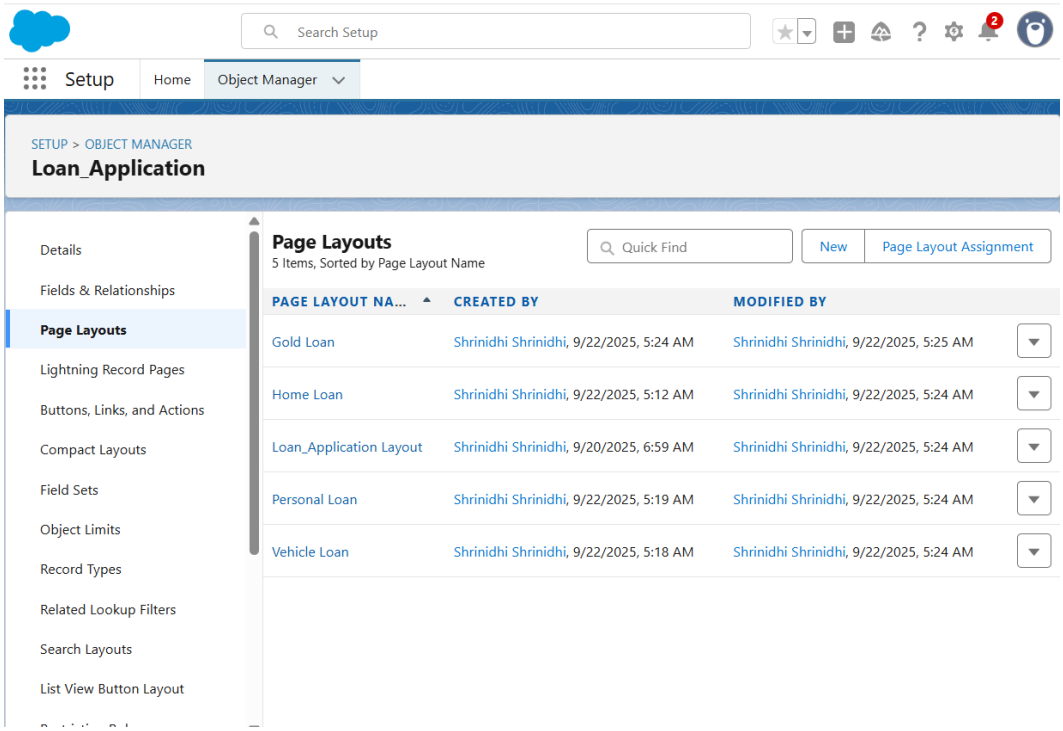


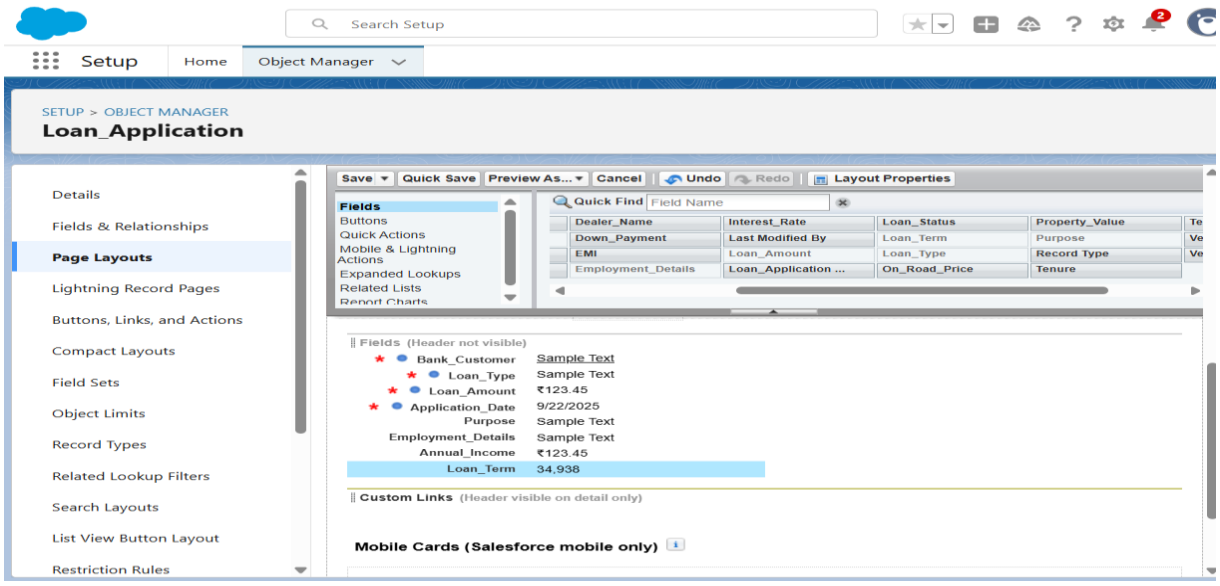

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Loan_Application
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Record Type
Personal Loan
Help for this Page
« Back to Custom Object: Loan_Application
Use the Edit button to change the properties of this record type. Use the Edit links in the Picklist Values related list to choose the picklist values available for records with this record type.
Edit
Record Type Label
Personal Loan
Active
Record Type Name
Personal_Loan
Namespace Prefix
Description
Created By
Shrinidhi Shrinidhi, 9/22/2025, 4:48 AM
Modified By
Shrinidhi Shrinidhi, 9/22/2025, 4:48 AM
Picklists Available for Editing
Picklists Available for Editing Help
Action
Field
Modified Date
Edit
Loan_Status
9/22/2025, 4:48 AM
Edit
Loan_Type
9/22/2025, 4:48 AM


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Loan_Application
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Record Type
Home Loan
Help for this Page
« Back to Custom Object: Loan_Application
Use the Edit button to change the properties of this record type. Use the Edit links in the Picklist Values related list to choose the picklist values available for records with this record type.
Edit
Record Type Label
Home Loan
Active
Record
Record Type: Home Loan ~ Salesforce - Developer Edition
Namespace Prefix
Description
Created By
Shrinidhi Shrinidhi, 9/22/2025, 4:45 AM
Modified By
Shrinidhi Shrinidhi, 9/22/2025, 4:45 AM
Picklists Available for Editing
Picklists Available for Editing Help
Action
Field
Modified Date
Edit
Loan_Status
9/22/2025, 4:45 AM
Edit
Loan_Type
9/22/2025, 4:45 AM



Page Layouts:

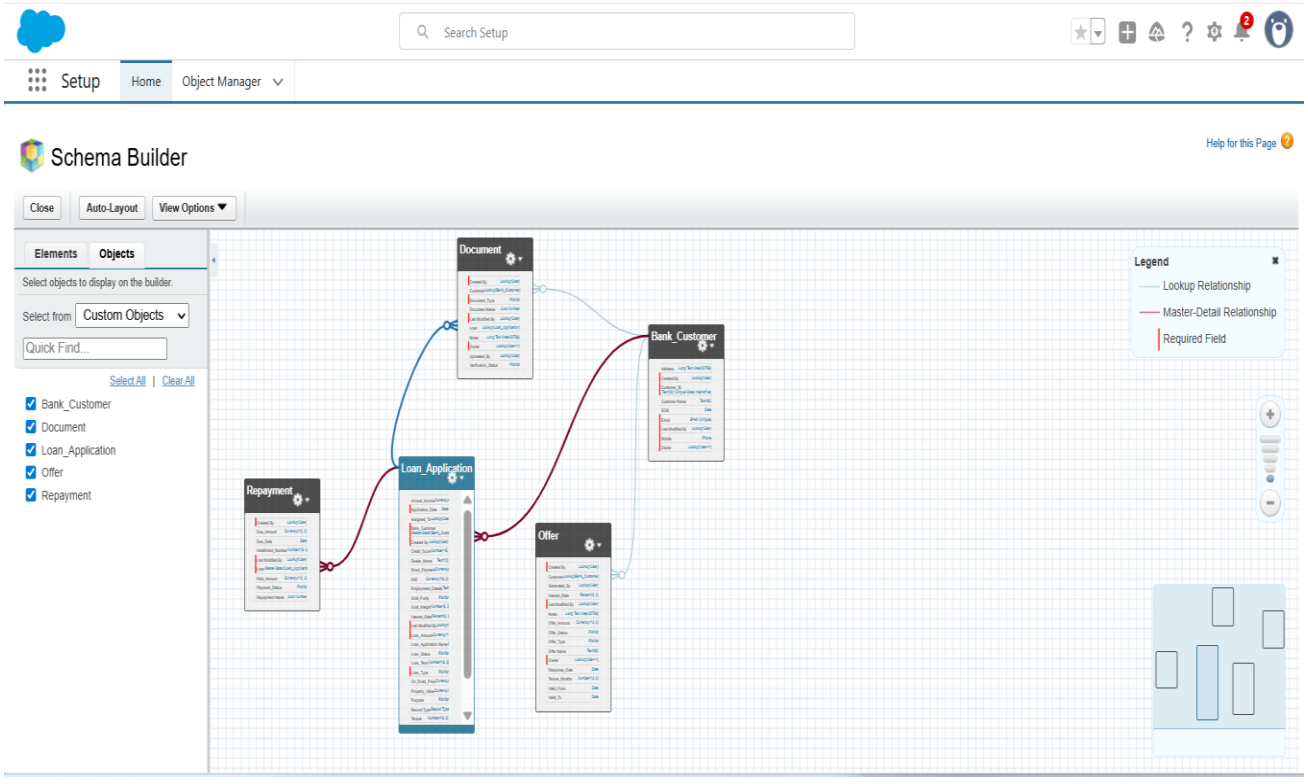




9. Schema Builder

What I did: I used Schema Builder to visualize relationships:
Bank_Customer__c (1) → Loan_Application__c (N) → Repayment__c (N)
Additionally, Loan_Application__c is related to Document__c and Offer__c.

Reason: Schema Builder provides a clear picture of how objects are connected, ensuring proper data relationships are maintained.



Phase 4: Process Automation (Admin)

Validation Rules, Scheduled Flow, and Approval Process

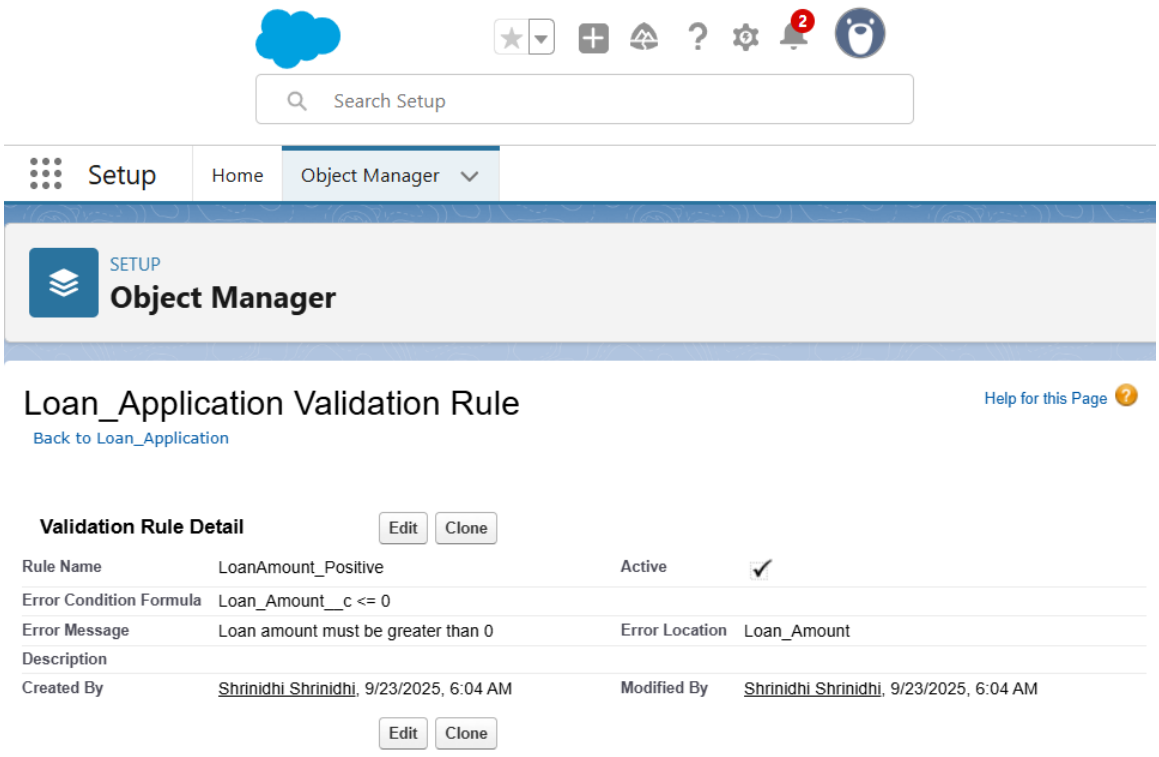
1. Validation Rule — Loan Amount > 0

What I did: I created a validation rule on Loan_Application__c object to ensure that Loan Amount is always greater than 0.

- Rule Name: LoanAmount_Positive
- Formula: Loan_Amount__c <= 0
- Error Message: Loan amount must be greater than 0.
- Error Location: Loan_Amount__c field.

Reason: This prevents users from entering zero or negative loan amounts, which are invalid in real-world banking operations.

Testing: Tried creating a Loan with Loan_Amount__c = 0 → System displayed the error message.

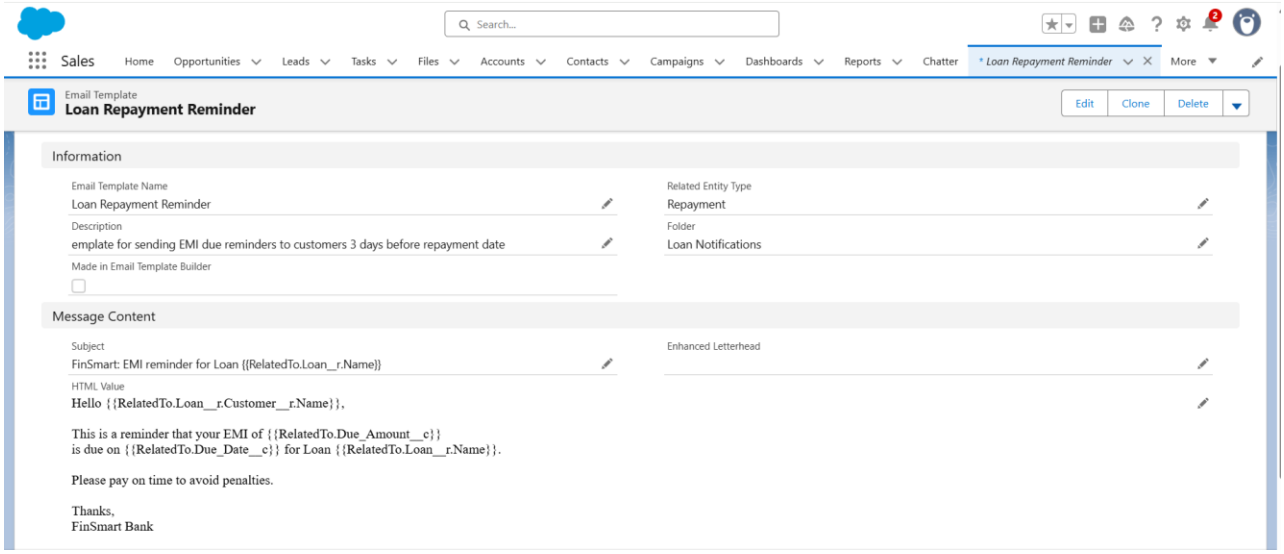


2. Scheduled Flow — EMI Reminder Emails

What I did: I built a Schedule-Triggered Flow to send EMI reminder emails to customers 3 days before due date.

Steps:

1. Created an Email Template named Loan Repayment Reminder.

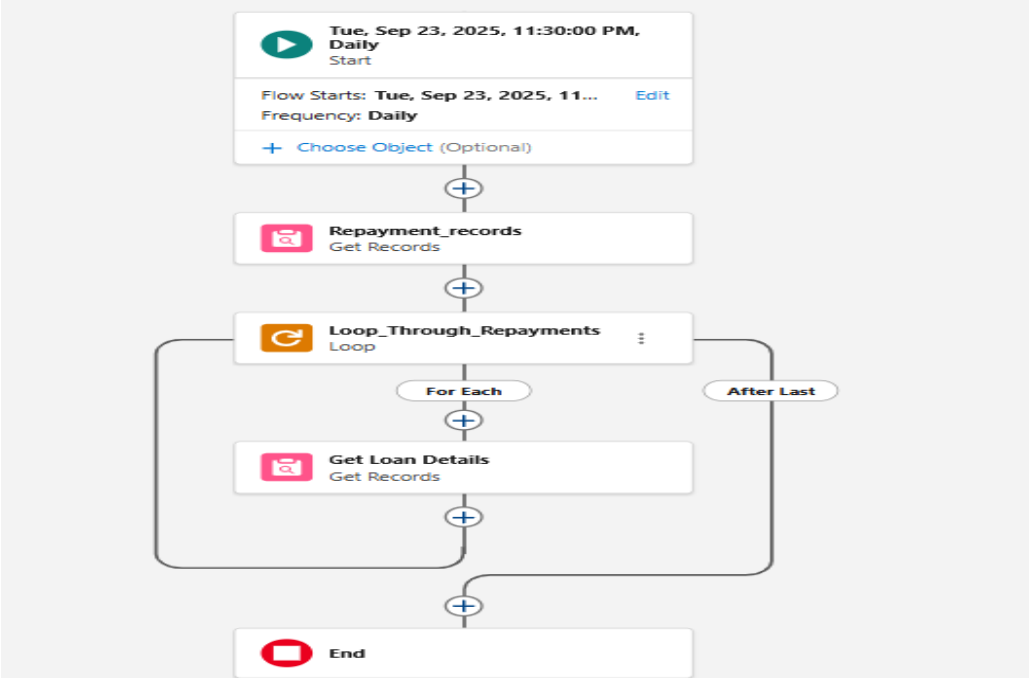


2. Built a Scheduled Flow:

- Frequency: Daily
- Formula Resource: targetDate = $Flow.CurrentDate + 3
- Get Records: Repayment__c where Due_Date__c = targetDate and Payment_Status__c ='Due'
- Loop through records and send email using Send Email action.

Reason: This ensures that customers are reminded about their EMI payments on time, reducing the chance of defaults and penalties.

Testing: For testing, I set targetDate = $Flow.CurrentDate + 0. Emails successfully triggered to test records.



3. Approval Process — High Value Loan Approvals

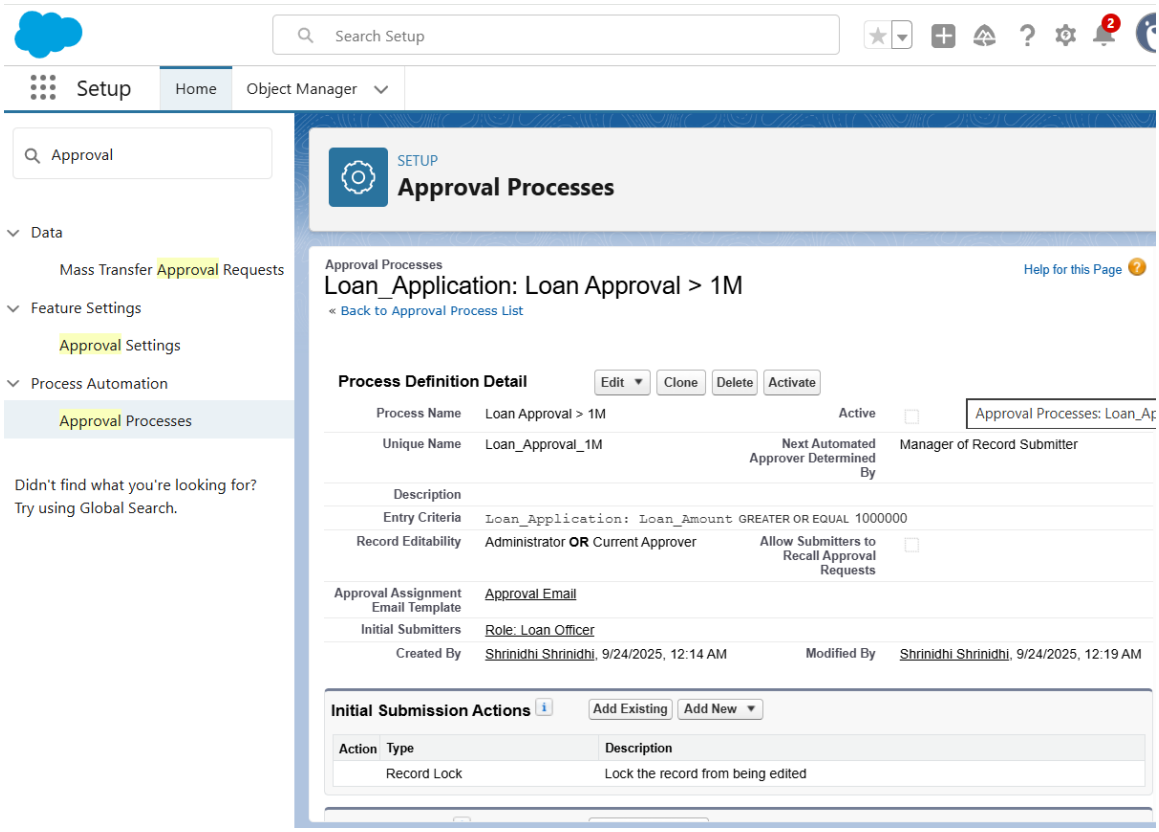
What I did: I created an Approval Process for Loan Applications where Loan Amount is >= 1,000,000.

Steps:

1. Object: Loan_Application__c
2. Entry Criteria: Loan_Amount__c >= 1000000
3. Approver: Manager of the Loan Owner (or specified role)
4. Initial Submission Action:
   • Field Update: Loan_Status__c = 'Under_Review'
   • Optional: Email alert to approver
5. Approval Steps: Approver reviews and approves/rejects
6. Final Approval Actions:
   • Field Update: Loan_Status__c = 'Approved'
   • Optional: Notification to submitter
   • Trigger repayment creation process (via flow)
7. Final Rejection Actions:
   • Field Update: Loan_Status__c = 'Rejected'
   • Email notification to applicant

Reason: This ensures that high-value loans go through a manager's approval before being processed, adding control and reducing risk.

Testing: Created a loan with Loan_Amount__c = 1,200,000. Submitted for Approval → Manager received request → Approved → Loan_Status__c updated to 'Approved'.



Summary

- Validation Rule ensures data accuracy (Loan Amount > 0).
- Scheduled Flow ensures timely communication (EMI reminders).
- Approval Process ensures governance and risk control for high-value loans.

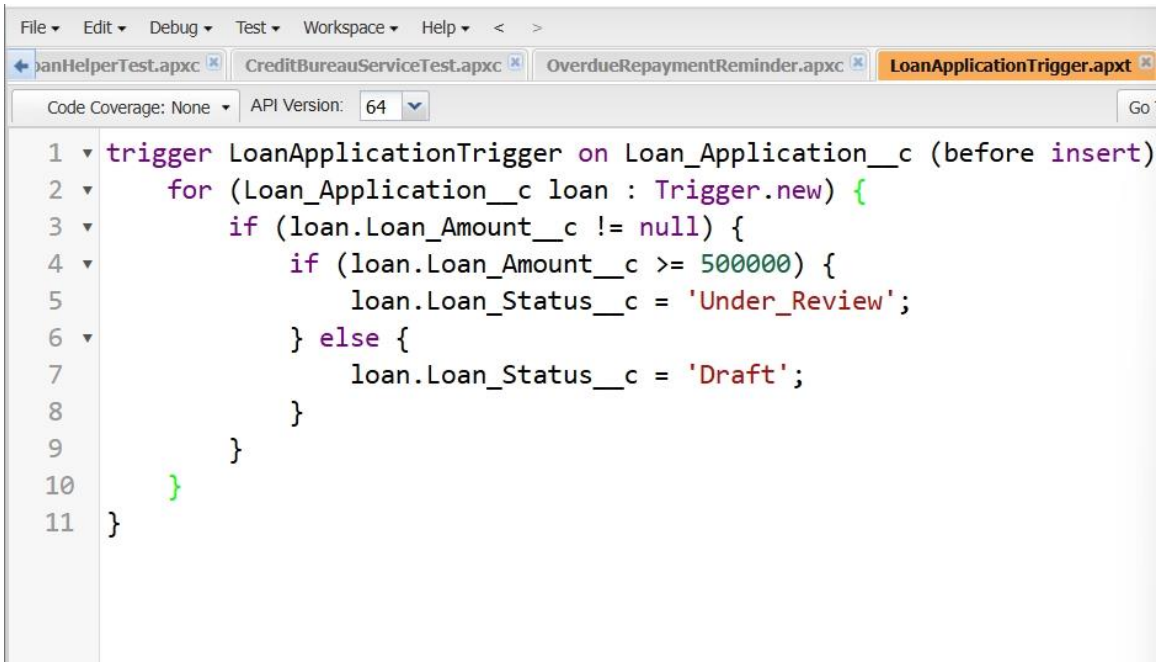
Phase 5: Apex Programming (Developer)

1: Create a Simple Trigger (Loan Application Auto Status)

- Purpose: Auto-set Loan_Status__c when a loan is created.
- What I did: Created an Apex Trigger on Loan_Application__c object to automatically set status based on Loan Amount.
- Reason: Ensures loan applications are automatically categorized as 'Draft' or 'Under_Review' depending on loan amount, reducing manual work and enforcing business rules.

Code:

```
trigger LoanApplicationTrigger on Loan_Application__c (before insert) {

for (Loan_Application__c loan : Trigger.new) {

      if (loan.Loan_Amount__c != null) {

         if (loan.Loan_Amount__c >= 500000) {

            loan.Loan_Status__c = 'Under_Review';

         } else {

            loan.Loan_Status__c = 'Draft';

         }

      }

   }

}
```



2: Create a Simple Apex Class (Helper for EMI)

- Purpose: Calculate EMI (basic formula).
- What I did: Created an Apex Class LoanHelper to calculate EMI values.
- Reason: Provides a reusable utility to compute EMI, useful for loan simulations and repayment planning.

Code:

```
public with sharing class LoanHelper {
    public static Decimal calculateEMI(Decimal amount, Decimal annualRate, Integer months) {
if (amount == null || annualRate == null || months == null || months <= 0) {
        return 0;
    }
    // Convert to Double for pow()
    Double principal = amount.doubleValue();
    Double monthlyRate = (annualRate.doubleValue() / 12) / 100;
    Double n = Double.valueOf(months);
```

```
        // Formula: EMI = P * r * (1+r)^n / ((1+r)^n - 1)
        Double powVal = Math.pow(1 + monthlyRate, n);
        Double emiDouble = (principal * monthlyRate * powVal) / (powVal - 1);

        // Cast back to Decimal
        Decimal emi = Decimal.valueOf(emiDouble);
        return emi.setScale(2);  // round to 2 decimal places
}
}
```

3: Simulate Credit Bureau Integration

- Purpose: Show external API call for Credit Score.
- What I did: Created a Named Credential, Apex Service class, and Mock class for integration testing.
- Reason: Simulates integration with external systems like Credit Bureau, demonstrating how Salesforce can handle callouts, responses, and use them to update loan application status.

3.1 Apex Class to Call Credit Bureau:

```
public with sharing class CreditBureauService {
@future(callout=true)
   public static void getCreditScore(Id loanId) {
      Loan_Application__c loan = [SELECT Id, Credit_Score__c FROM Loan_Application__c
WHERE Id = :loanId LIMIT 1];

      HttpRequest req = new HttpRequest();
      req.setEndpoint('callout:NamedCredential_CreditBureau/score?cust=' + loan.Id);
req.setMethod('GET');

      Http http = new Http();
      HttpResponse res = http.send(req);
```

```
        if (res.getStatusCode() == 200) {
            Integer score = Integer.valueOf(res.getBody());
            loan.Credit_Score__c = score;
            if (score >= 650) loan.Loan_Status__c = 'Under_Review';
else loan.Loan_Status__c = 'Rejected';
            update loan;
        }
    }
}
```

3.2 Mock Class for Testing:

```
@IsTest
global class CreditBureauMock implements HttpCalloutMock {
global HttpResponse respond(HttpRequest req) {
        HttpResponse res = new HttpResponse();
        res.setStatusCode(200);
        res.setBody('720'); // fake credit score
        return res;
    }
}
```

4: Create Test Classes

- Purpose: Ensure >75% coverage and validate logic.
- What I did: Wrote test classes for LoanHelper and CreditBureauService to validate functionality and meet Salesforce coverage requirements.
- Reason: Test classes are mandatory in Salesforce to deploy Apex code and ensure system reliability.

LoanHelperTest.cls:

```
@isTest
private class LoanHelperTest {
    @isTest
```

```
    static void testLoanApplicationTriggerAndEMI() {
        Bank_Customer__c cust = new Bank_Customer__c(Name='Test Cust',
Email__c='test@test.com');
        insert cust;

        Loan_Application__c loan = new Loan_Application__c(
Bank_Customer__c = cust.Id,
            Loan_Amount__c = 600000,
            Interest_Rate__c = 10,
            Tenure_Months__c = 12
        );
        insert loan; // trigger runs

        Loan_Application__c inserted = [SELECT Loan_Status__c FROM Loan_Application__c
WHERE Id=:loan.Id];
        System.assertEquals('Under_Review', inserted.Loan_Status__c);

        Decimal emi = LoanHelper.calculateEMI(100000, 12, 12);
System.assert(emi > 0, 'EMI should be positive');
    }
}
```

CreditBureauServiceTest.cls:

```
@isTest
private class CreditBureauServiceTest {
    @isTest
    static void testCreditScoreUpdate() {
        Bank_Customer__c cust = new Bank_Customer__c(Name='Cust B',
Email__c='cust@test.com');
```

```
        insert cust;

        Loan_Application__c loan = new Loan_Application__c(
Bank_Customer__c = cust.Id,
            Loan_Amount__c = 400000,
            Loan_Status__c = 'Submitted'
        );
        insert loan;

        // mock callout
        Test.setMock(HttpCalloutMock.class, new CreditBureauMock());

        Test.startTest();
        CreditBureauService.getCreditScore(loan.Id);
Test.stopTest();

        Loan_Application__c updated = [SELECT Credit_Score__c, Loan_Status__c FROM
Loan_Application__c WHERE Id = :loan.Id];
        System.assertEquals(720, updated.Credit_Score__c);
        System.assertEquals('Under_Review', updated.Loan_Status__c);
}
}
```

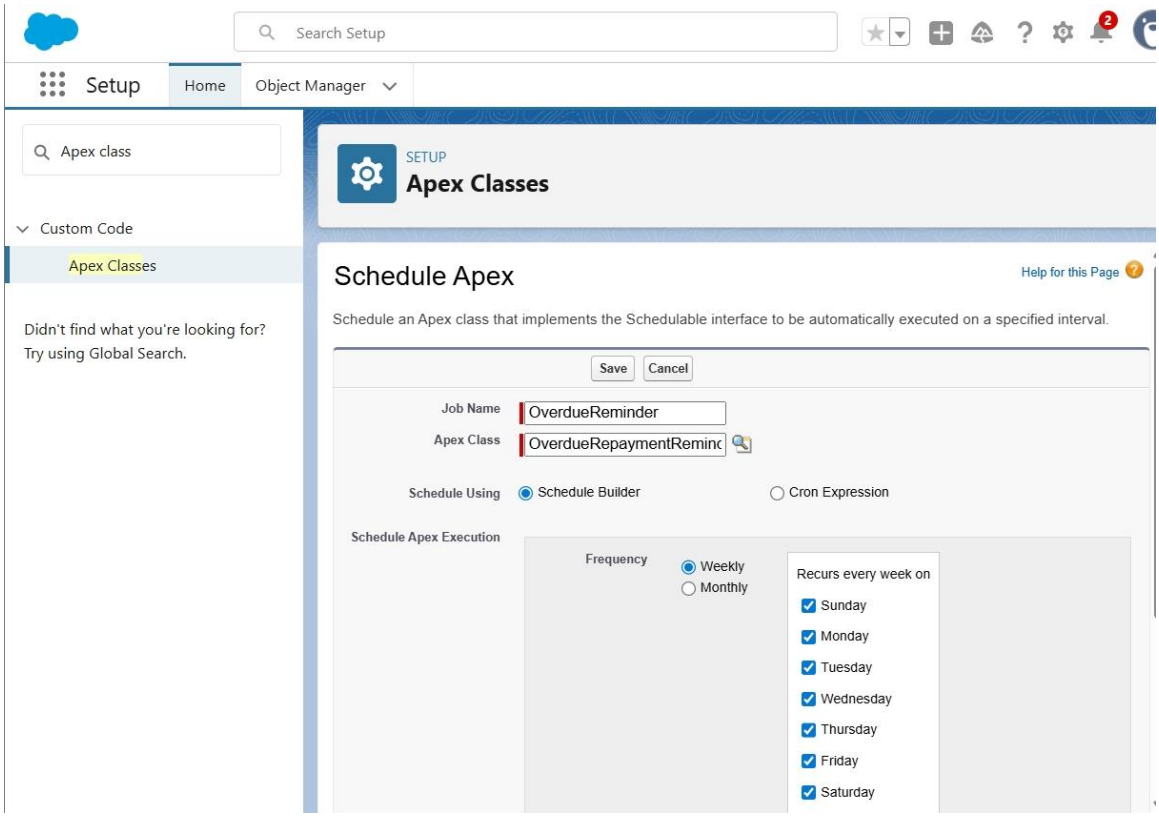
5: Simple Scheduled Apex

- Purpose: Simple scheduled check for overdue repayments.
- What I did: Created a schedulable class that fetches overdue repayments and logs them.
- Reason: Demonstrates ability to schedule background jobs in Salesforce to monitor repayment compliance.

Code:

```
public with sharing class OverdueRepaymentReminder implements Schedulable {

   public void execute(SchedulableContext sc) {

      List<Repayment__c> overdues = [

         SELECT Id, Loan__c, Due_Amount__c

         FROM Repayment__c

         WHERE Payment_Status__c = 'Overdue'

];

      for (Repayment__c r : overdues) {

         System.debug('Overdue repayment: ' + r.Id + ' Amount: ' + r.Due_Amount__c);

}

   }

}
```

To schedule: Setup → Apex Classes → Schedule Apex → Job Name = OverdueReminder, Class = OverdueRepaymentReminder, set time.


Search Setup
Setup
Home
Object Manager
Apex class
Custom Code
Apex Classes
Didn't find what you're looking for?
Try using Global Search.
SETUP
Apex Classes
Schedule Apex
Help for this Page
Schedule an Apex class that implements the Schedulable interface to be automatically executed on a specified interval.
Save
Cancel
Job Name
OverdueReminder
Apex Class
OverdueRepaymentReminc
Schedule Using
Schedule Builder
Cron Expression
Schedule Apex Execution
Frequency
Weekly
Monthly
Recurs every week on
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

Phase 6: User Interface Development

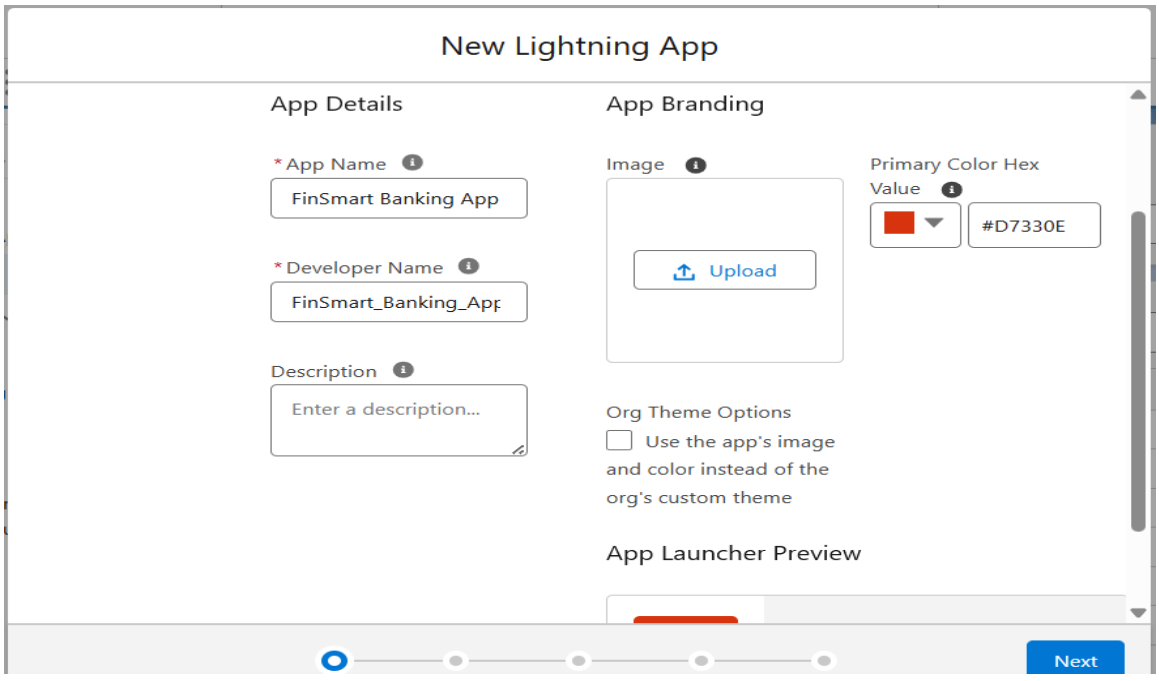
1: Create Lightning App

What I did: I created a new Lightning App using App Manager in Salesforce to organize all loan-related features in one application.

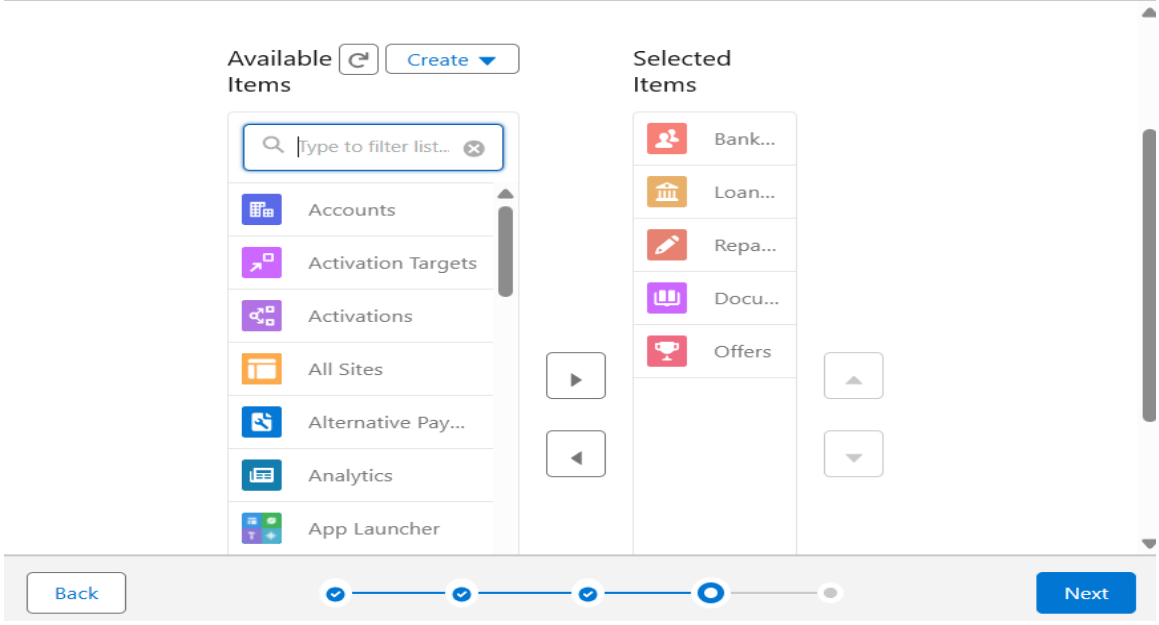
Reason: A Lightning App provides a dedicated workspace for users. By grouping related objects and features (Customers, Loan Applications, Repayments, Documents, Reports), it improves navigation and user experience.

Steps:

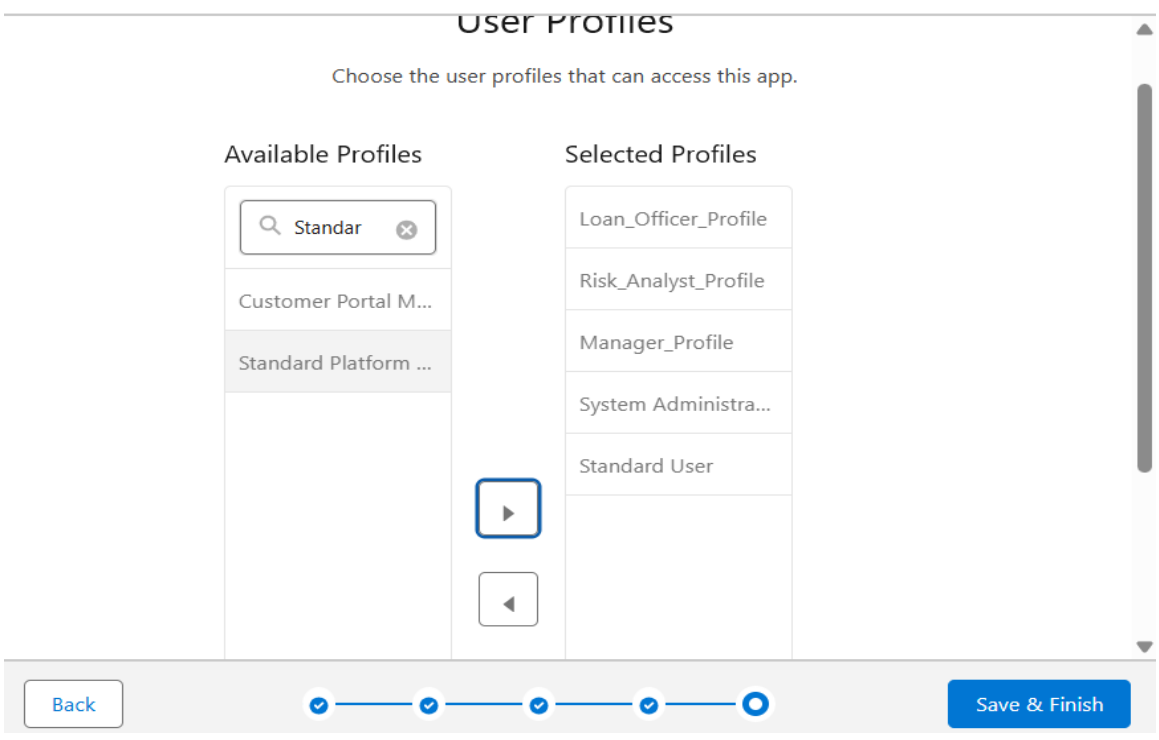
1. Go to Setup → App Manager → New Lightning App.
2. Enter details:
   • App Name: FinSmart Banking App
   • Navigation Type: Standard (Lightning)
   • Optionally upload a logo → Next
3. Add Tabs:
   • Customers
   • Loan Applications
   • Repayments
   • Documents
   • Reports
4. Save & Finish.



New Lightning App



New Lightning App



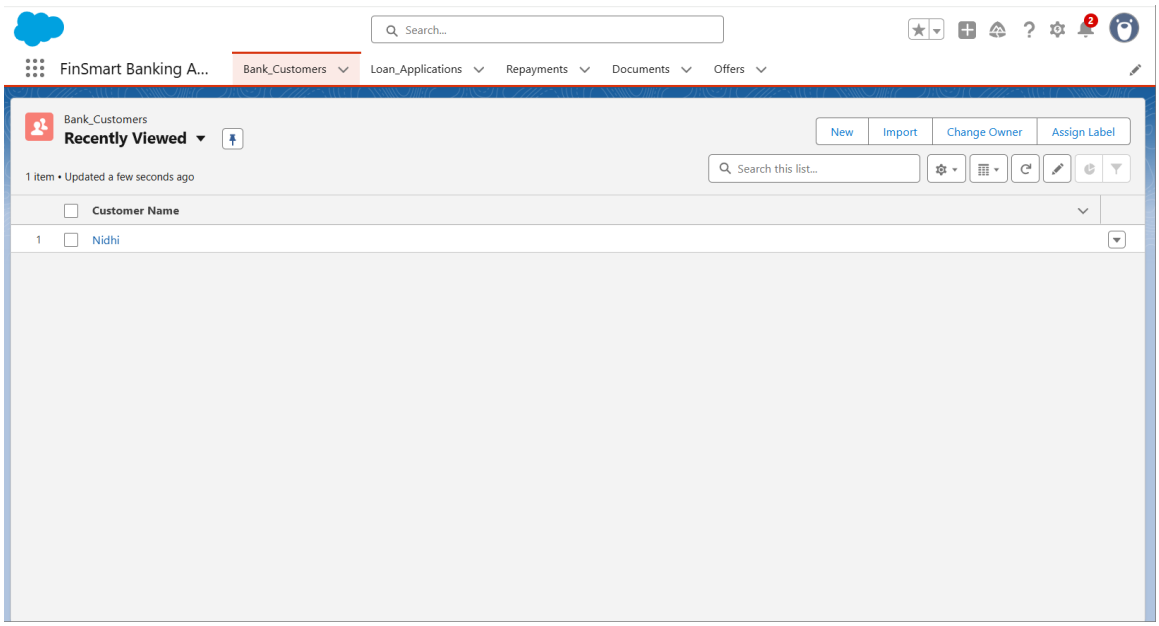
Step 2: Confirm Tabs in App Launcher

What I did: I verified the app setup by launching it from the App Launcher and ensuring all required tabs are visible.

Reason: Verification ensures the app is properly configured and that users will be able to access all necessary components in their daily work.

Steps:

1. Go to Setup → App Launcher.
2. Open the newly created FinSmart Banking App.
3. Confirm that all required tabs (Customers, Loan Applications, Repayments, Documents, Reports) are visible.



Summary

• What I did: Created a dedicated Lightning App named FinSmart Banking App and added navigation tabs for core objects.
• Reason: To provide a centralized, easy-to-use application that helps users manage customers, loans, repayments, documents, and reports efficiently.

Next step: Add screenshots of app creation and tab confirmation for documentation.

Phase 7: Integration & External Access

Credit Bureau Integration

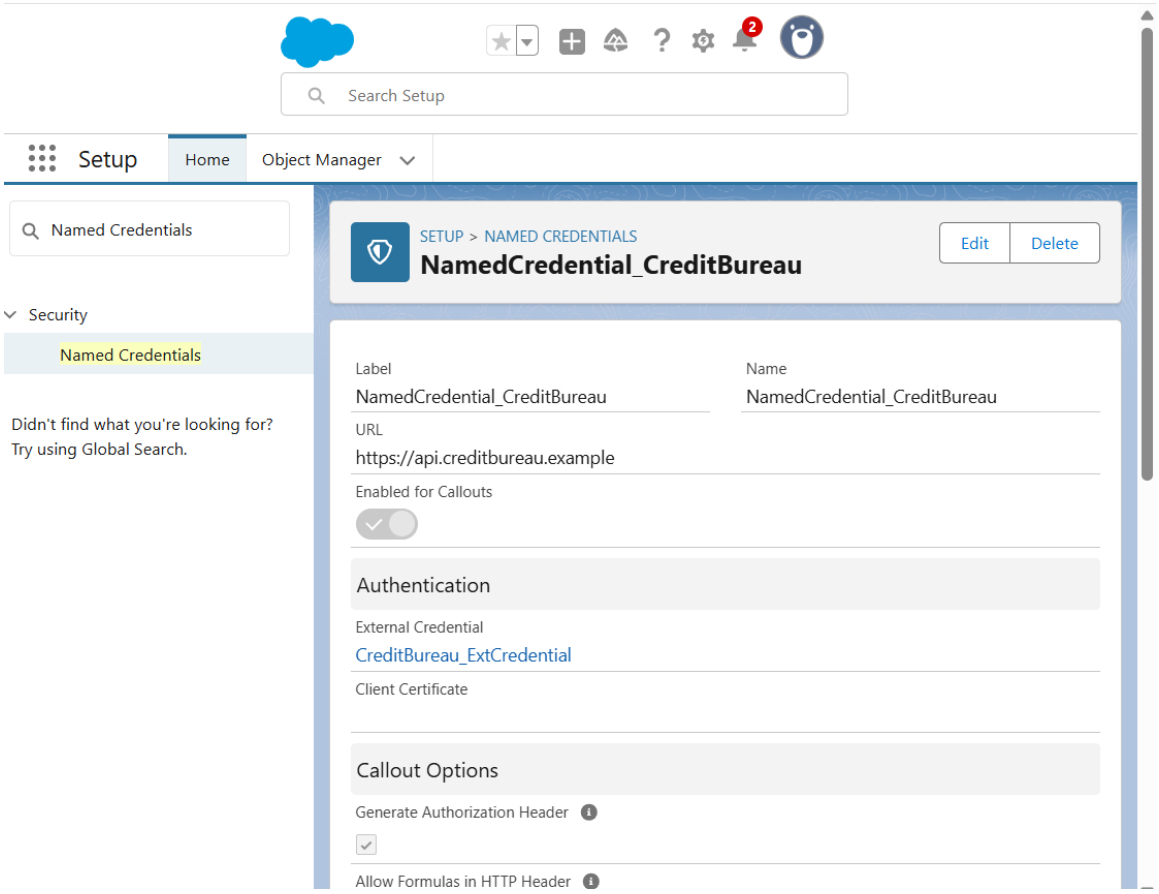
Step 1: Named Credential

What I did: I created a Named Credential in Salesforce for connecting to the Credit Bureau API.

Reason: Named Credentials simplify API callouts by securely storing the endpoint URL and authentication details. This avoids hardcoding sensitive information in Apex classes and makes future updates easier.

Steps:

1. Go to Setup → Named Credentials → New.



Apex Callout:

public with sharing class CreditBureauService {

@future(callout=true)

```
    public static void getCreditScore(Id loanId) {
        Loan_Application__c loan = [SELECT Id, Credit_Score__c FROM Loan_Application__c
WHERE Id = :loanId LIMIT 1];

        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:NamedCredential_CreditBureau/score?cust=' + loan.Id);
        req.setMethod('GET');

        Http http = new Http();
        HttpResponse res = http.send(req);

        if (res.getStatusCode() == 200) {
            Integer score = Integer.valueOf(res.getBody());
            loan.Credit_Score__c = score;
            if (score >= 650) loan.Loan_Status__c = 'Under_Review';
            else loan.Loan_Status__c = 'Rejected';
            update loan;
        }
    }
}
```

Step 2: Test Callout (Using HttpCalloutMock)

What I did: I created a Mock class and a Test class to simulate the Credit Bureau API response without calling a real service.

Reason: Salesforce requires test classes to mock callouts. This ensures code can be deployed while avoiding dependencies on external systems during testing.

Mock Class Code:

```
@IsTest
global class CreditBureauMock implements HttpCalloutMock {
```

```
    global HttpResponse respond(HttpRequest req) {
        HttpResponse res = new HttpResponse();
        res.setStatusCode(200);
        res.setBody('720'); // fake credit score
        return res;
    }
}
```

Test Class Code:

```
@isTest
private class CreditBureauServiceTest {
    @isTest
    static void testCreditScoreUpdate() {
        Bank_Customer__c cust = new Bank_Customer__c(Name='Cust B',
Email__c='cust@test.com');
        insert cust;

        Loan_Application__c loan = new Loan_Application__c(
            Bank_Customer__c = cust.Id,
            Loan_Amount__c = 400000,
            Loan_Status__c = 'Submitted'
        );
        insert loan;

        // mock callout
        Test.setMock(HttpCalloutMock.class, new CreditBureauMock());

        Test.startTest();
```

```
        CreditBureauService.getCreditScore(loan.Id);

        Test.stopTest();


        Loan_Application__c updated = [SELECT Credit_Score__c, Loan_Status__c FROM
Loan_Application__c WHERE Id = :loan.Id];

        System.assertEquals(720, updated.Credit_Score__c);

        System.assertEquals('Under_Review', updated.Loan_Status__c);

    }

}
```

Summary

• What I did: Configured a Named Credential and implemented a minimal Apex callout setup with test classes.
• Reason: This ensures Salesforce can simulate Credit Bureau integration securely and pass deployment requirements without relying on a real API.

Next step: Add screenshots of Named Credential setup and test execution for documentation

Phase 8: Data Management & Deployment

Data Import & Duplicate Management (Step by Step Procedure)

Preparation — Create External ID Fields

What I did: I created External ID fields on Customer__c to ensure clean parent–child relationships during CSV imports.

Reason: External ID fields allow Salesforce to match and link records using unique values from CSVs. This is important when importing data with relationships (e.g., Loans linked to Customers).

Steps:

1. Setup → Object Manager → Customer__c → Fields & Relationships → New → Text (Length 20).
2. Label: Customer ID, Field Name: Customer_ID__c.
3. Mark as External ID and Unique (optional) → Save.

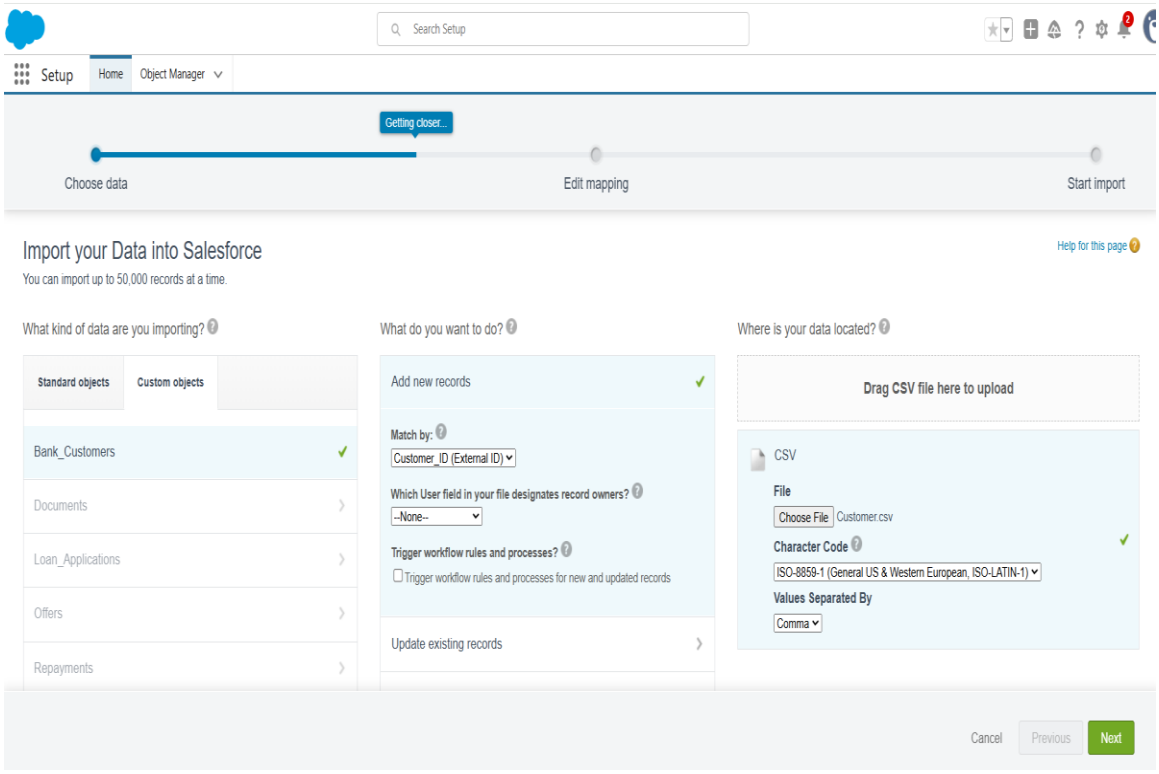
1: Import Data using Data Import Wizard

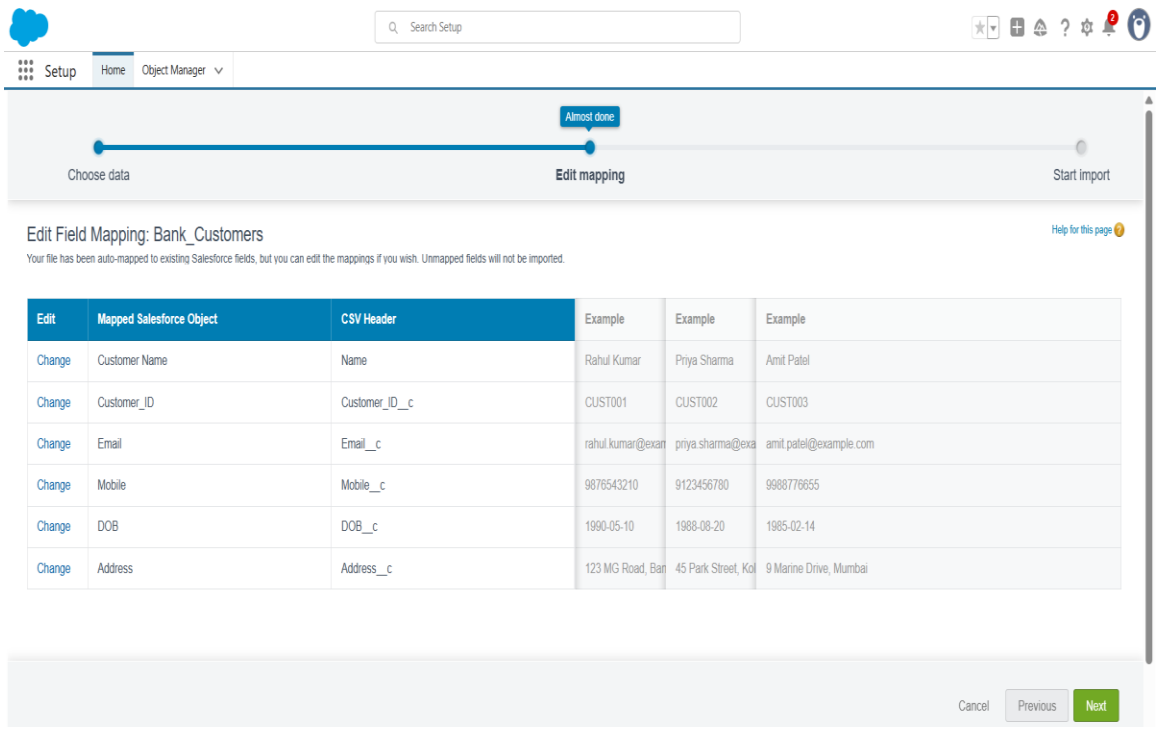
What I did: I used Salesforce Data Import Wizard to upload Customer data from CSV.

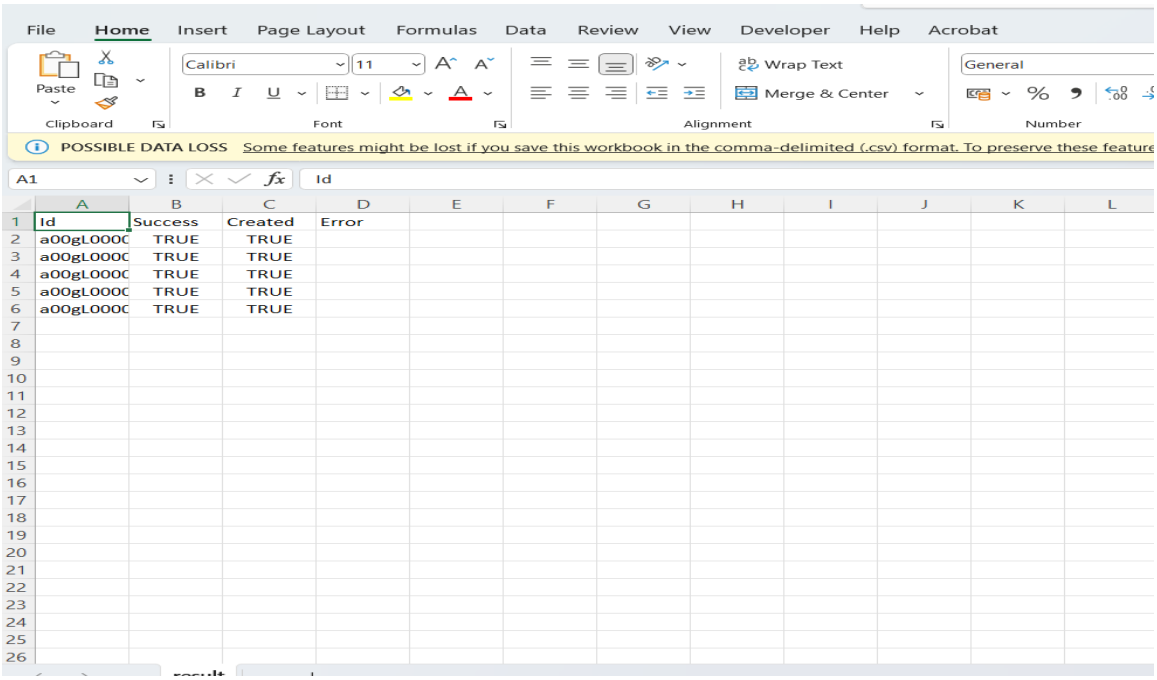
Reason: The Data Import Wizard is simple, user-friendly, and ideal for small datasets. It automatically maps fields when column headers match API field names.

Steps:

1. Setup → Data Import Wizard → Launch Wizard.
2. For Customers:
   - Select Custom Objects → choose Customer (Customer__c).
   - Upload Customer.csv.
   - Map fields if required (auto-map works when names match).
   - Start Import → Wait → Review results.


Search Setup
Setup
Home
Object Manager
Getting closer...
Choose data
Edit mapping
Start import
Import your Data into Salesforce
You can import up to 50,000 records at a time.
Help for this page
What kind of data are you importing?
What do you want to do?
Where is your data located?
Standard objects
Custom objects
Bank_Customers
Documents
Loan_Applications
Offers
Repayments
Add new records
Match by:
Customer_ID (External ID)
Which User field in your file designates record owners?
--None--
Trigger workflow rules and processes?
Trigger workflow rules and processes for new and updated records
Update existing records
Drag CSV file here to upload
CSV
File
Choose File
Customer.csv
Character Code
ISO-8859-1 (General US & Western European, ISO-LATIN-1)
Values Separated By
Comma
Cancel
Previous
Next


Search Setup
Setup
Home
Object Manager
Almost done
Choose data
Edit mapping
Start import
Edit Field Mapping: Bank_Customers
Your file has been auto-mapped to existing Salesforce fields, but you can edit the mappings if you wish. Unmapped fields will not be imported.
Help for this page
Edit
Mapped Salesforce Object
CSV Header
Example
Example
Example
Change
Customer Name
Name
Rahul Kumar
Priya Sharma
Amit Patel
Change
Customer_ID
Customer_ID__c
CUST001
CUST002
CUST003
Change
Email
Email__c
rahul.kumar@exam
priya.sharma@exa
amit.patel@example.com
Change
Mobile
Mobile__c
9876543210
9123456780
9988776655
Change
DOB
DOB__c
1990-05-10
1988-08-20
1985-02-14
Change
Address
Address__c
123 MG Road, Ban
45 Park Street, Kol
9 Marine Drive, Mumbai
Cancel
Previous
Next



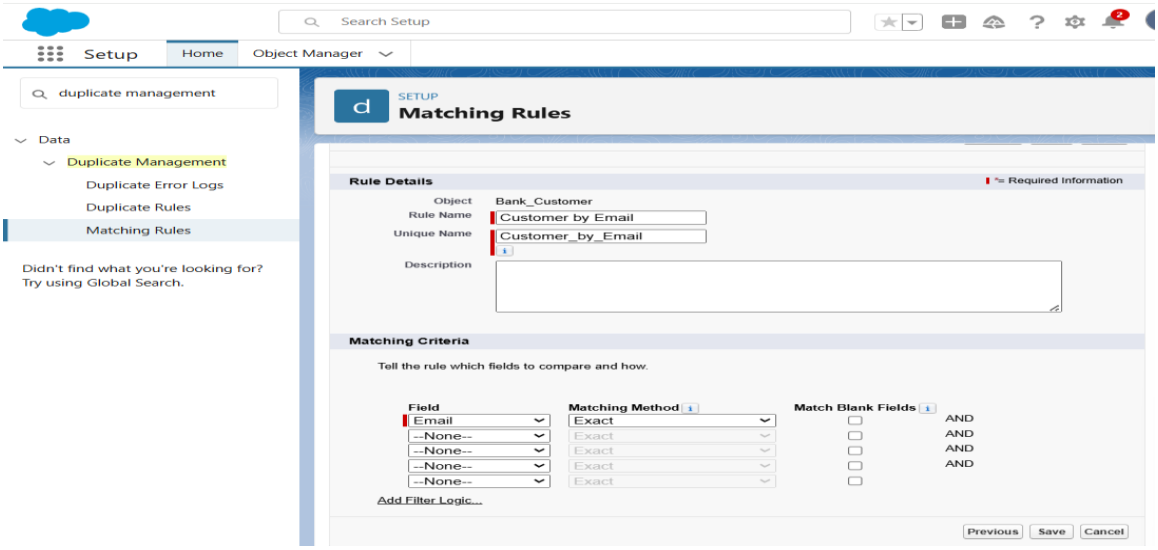
2: Duplicate Management

What I did: I created Matching Rules and Duplicate Rules to prevent duplicate customer records.

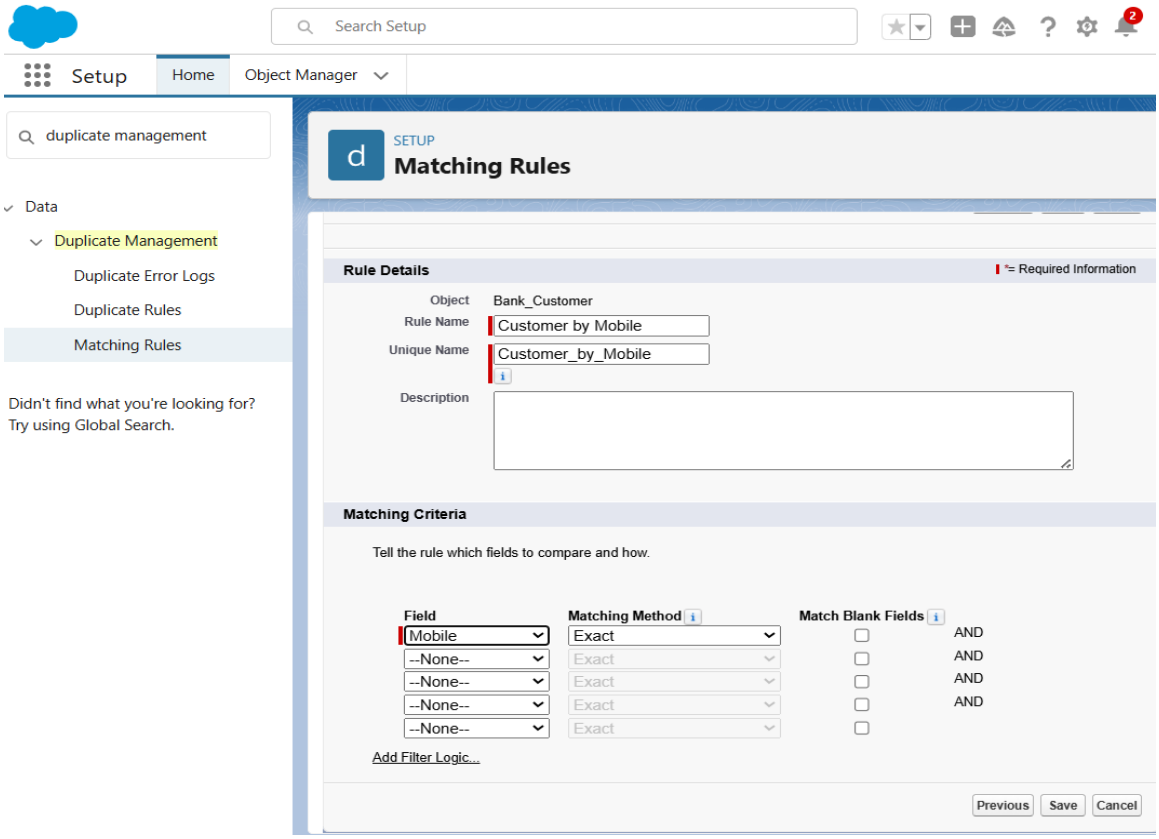
Reason: Duplicate management ensures data quality by preventing multiple entries for the same customer based on email or mobile number.

Steps:

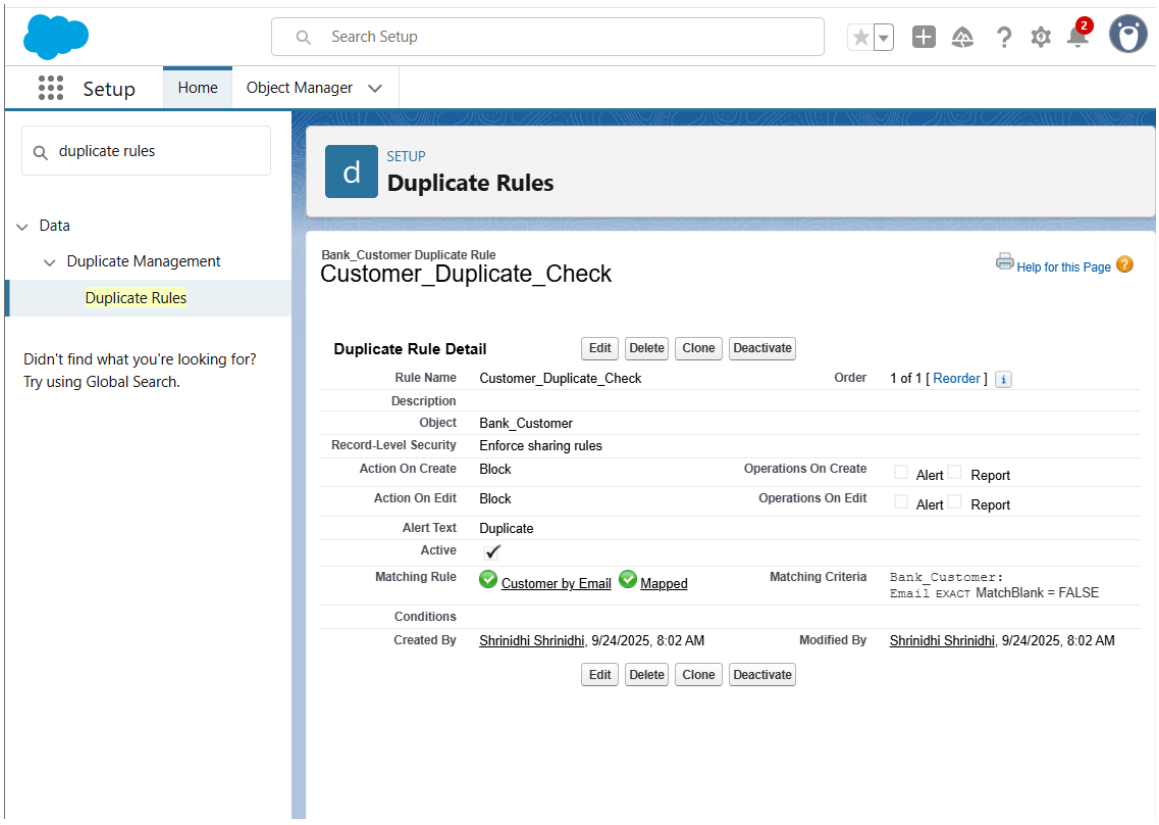
1. Setup → Duplicate Management → **Matching Rules.**
   • Customer by Email: New Matching Rule → Object: Customer → add field Email__c (fuzzy/exact match) → Save & Activate.



• Customer by Mobile: New Matching Rule → Object: Customer → add field Mobile__c → Save & Activate.



2. Setup → **Duplicate Rules** → New Rule for Customer.
   • Choose Object: Customer__c.
   • Use matching rules above (Email, Mobile).
   • Action: Block or Alert when duplicates detected.
   • Activate.
3. When re-importing data, duplicates are flagged.


Search Setup
Setup
Home
Object Manager
duplicate rules
Data
Duplicate Management
Duplicate Rules
Didn't find what you're looking for?
Try using Global Search.
SETUP
Duplicate Rules
Bank_Customer Duplicate Rule
Customer_Duplicate_Check
Help for this Page
Duplicate Rule Detail
Edit
Delete
Clone
Deactivate
Rule Name
Customer_Duplicate_Check
Order
1 of 1 [Reorder]
Description
Object
Bank_Customer
Record-Level Security
Enforce sharing rules
Action On Create
Block
Operations On Create
Alert
Report
Action On Edit
Block
Operations On Edit
Alert
Report
Alert Text
Duplicate
Active
Matching Rule
Customer by Email
Mapped
Matching Criteria
Bank_Customer:
Email EXACT MatchBlank = FALSE
Conditions
Created By
Shrinidhi Shrinidhi, 9/24/2025, 8:02 AM
Modified By
Shrinidhi Shrinidhi, 9/24/2025, 8:02 AM
Edit
Delete
Clone
Deactivate

Phase 9: Reporting, Dashboards & Security Review

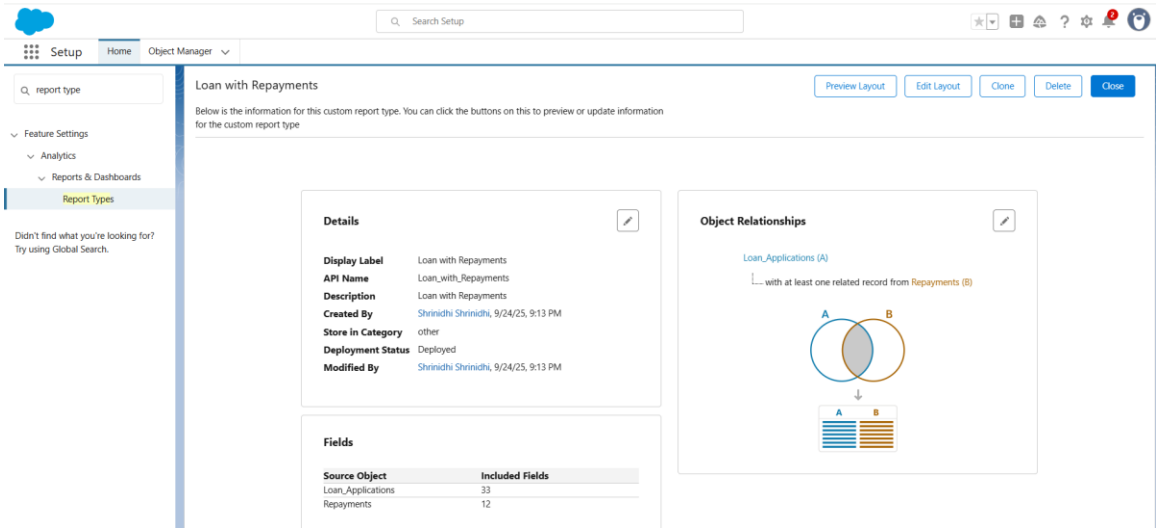
Step 1: Create Custom Report Types

What I did: I created custom report types to combine related objects for reporting.

Reason: Custom Report Types allow users to generate insights that span multiple objects, such as Customers with their Loans or Loans with their Repayments.

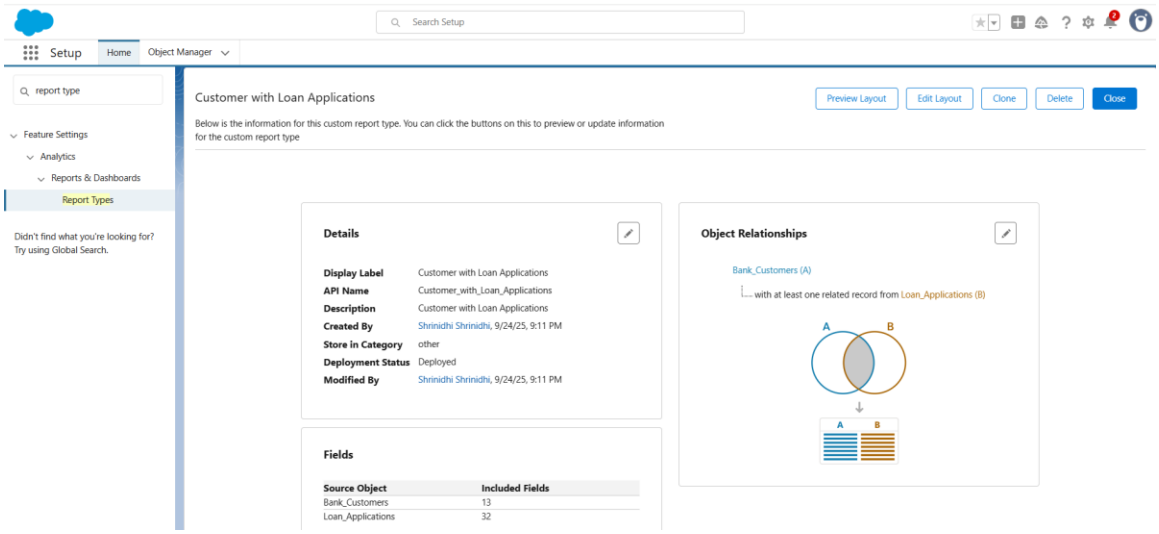
Steps:

1. Go to Setup → Report Types → New Custom Report Type.
2. Customer with Loan Applications:
   • Primary Object: Customer__c
   • Child Object: Loan_Application__c (Each ‘A’ record must have at least one related ‘B’ record)
   • Save & Deploy.



3. Loan with Repayments:
   • Primary Object: Loan_Application__c
   • Child Object: Repayment__c
   • Save & Deploy.

Outcome: These report types enable building combined reports.

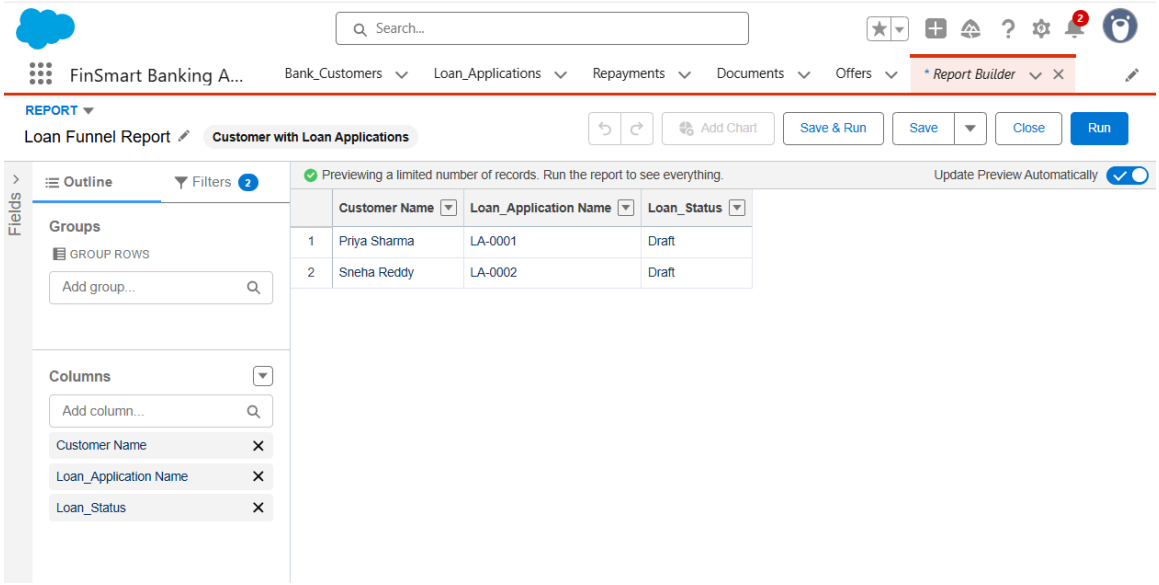


Step 2: Create Reports

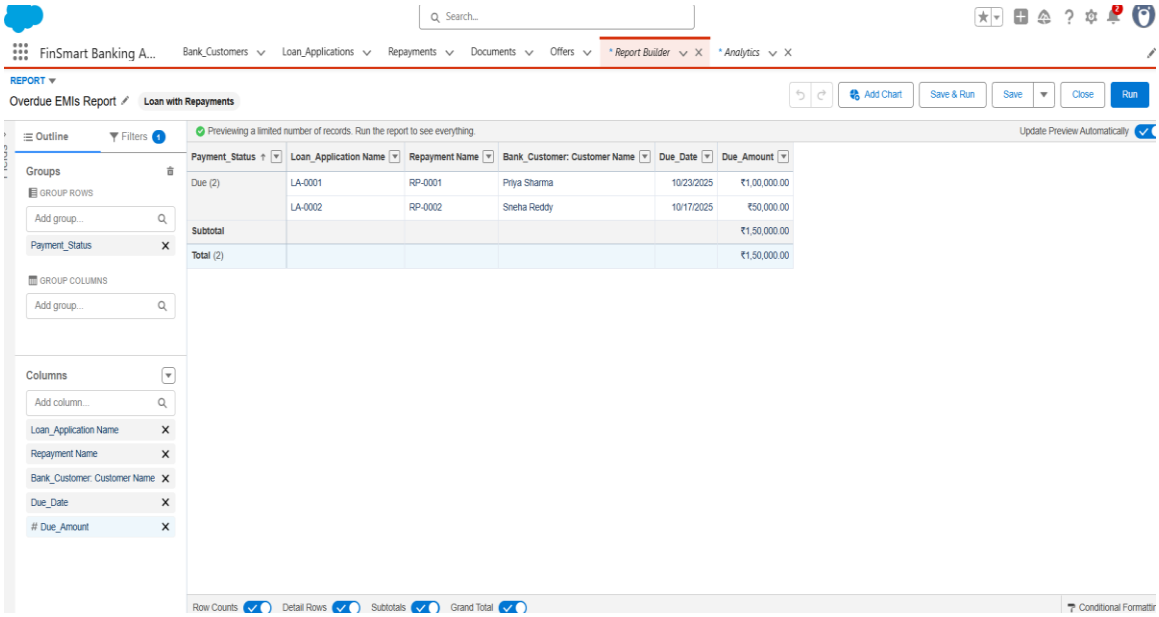
What I did: I created reports using the custom report types for business insights.

Reason: Reports summarize data and provide detailed views for decision-making.

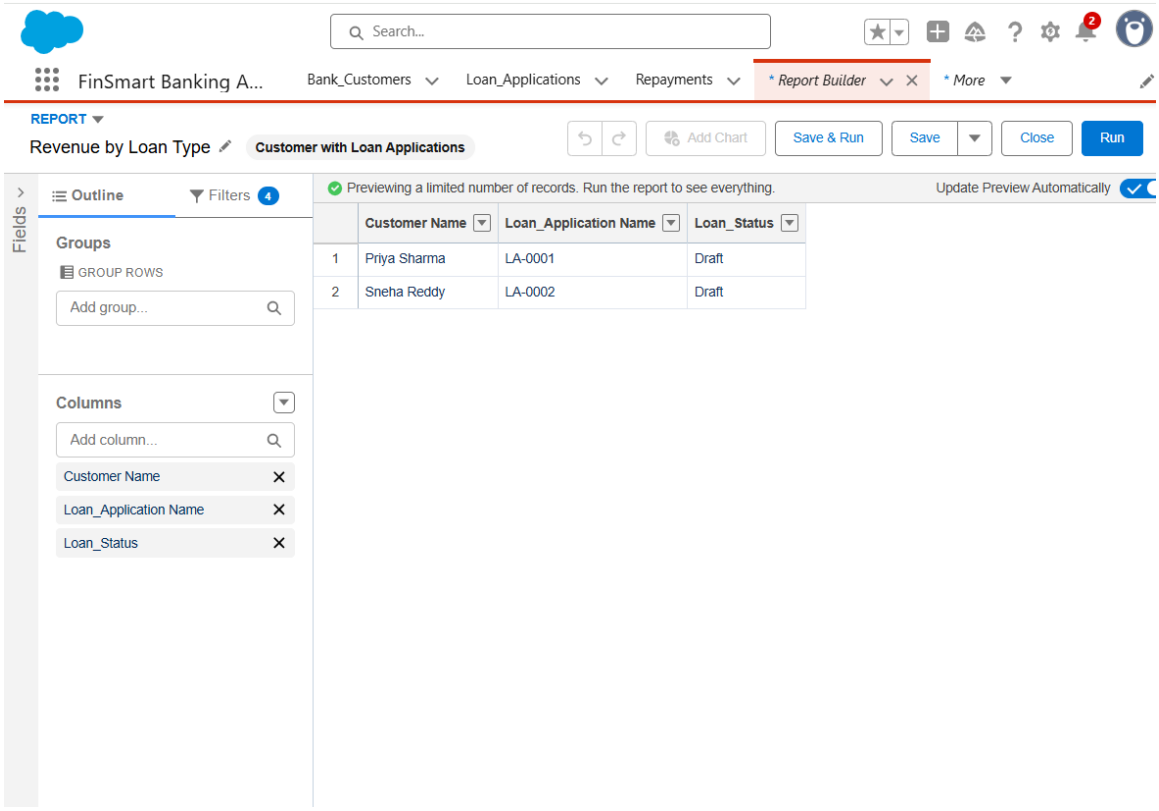
1. Loan Funnel:
   - Report Type: Customer with Loan Applications
   - Group by Loan_Status__c
   - Columns: Customer Name, Loan Amount, Status
   - Save as Loan Funnel Report.



2. Overdue EMIs:
   • Report Type: Loan with Repayments
   • Filter: Payment_Status__c = Overdue
   • Columns: Loan Name, Customer, Due Date, Amount
   • Save as Overdue EMIs Report.



3. Revenue by Loan Type:
   • Report Type: Customer with Loan Applications
   • Filter: Loan_Status__c = Disbursed
   • Summarize Loan_Amount__c by Loan_Type__c
   • Save as Revenue by Loan Type Report.



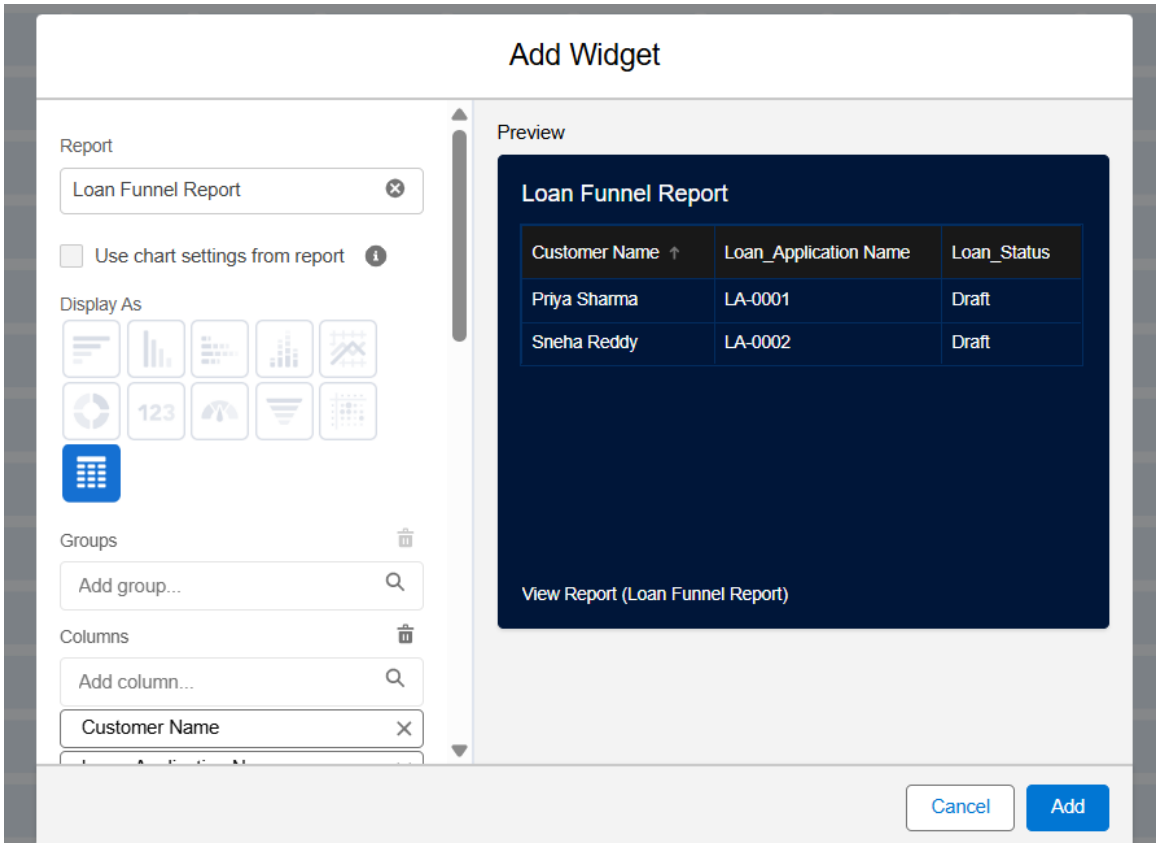
Step 3: Create Dashboard

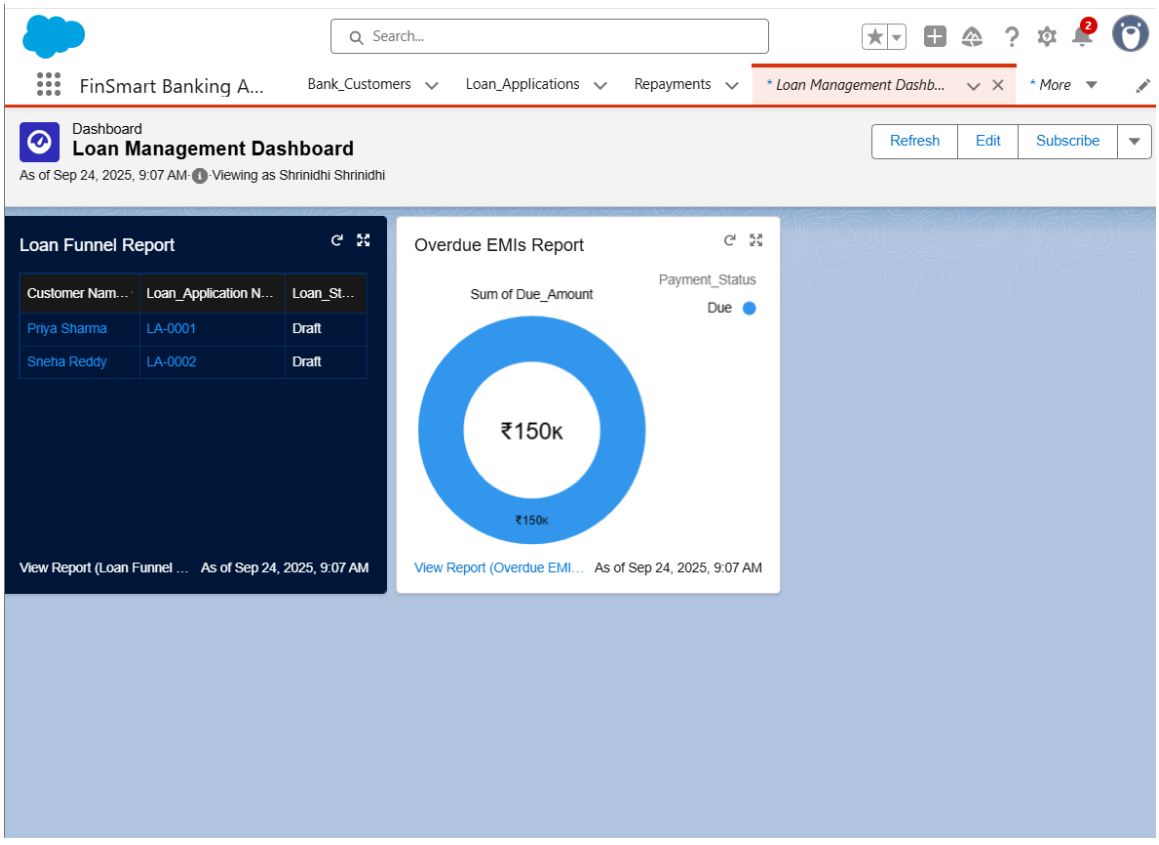
What I did: I created a dashboard to visualize loan performance and repayment trends.

Reason: Dashboards provide management with quick access to KPIs and visual trends for better decision-making.

Steps:

1. Go to Dashboards → New Dashboard.
2. Name: Loan Management Dashboard, choose a folder.
3. Add Components:
   • KPI Tiles:
   - Total Disbursed Loans
   - Overdue Count
   - Average Credit Score
   • Charts:
   - Loans by Type → Pie/Bar chart
   - Overdue Trend → Line chart
4. Configure components to use reports from Step 2.
5. Save & Refresh Dashboard.


Add Widget
Report
Loan Funnel Report
Use chart settings from report
Display As
Groups
Add group...
Columns
Add column...
Customer Name
Preview
Loan Funnel Report
Customer Name
Loan_Application Name
Loan_Status
Priya Sharma
LA-0001
Draft
Sneha Reddy
LA-0002
Draft
View Report (Loan Funnel Report)
Cancel
Add


Search...
FinSmart Banking A...
Bank_Customers
Loan_Applications
Repayments
* Loan Management Dashb...
* More
Dashboard
Loan Management Dashboard
As of Sep 24, 2025, 9:07 AM · Viewing as Shrinidhi Shrinidhi
Refresh
Edit
Subscribe
Loan Funnel Report
Customer Nam...
Loan_Application N...
Loan_St...
Priya Sharma
LA-0001
Draft
Sneha Reddy
LA-0002
Draft
View Report (Loan Funnel ...
As of Sep 24, 2025, 9:07 AM
Overdue EMIs Report
Sum of Due_Amount
Payment_Status
Due
₹150K
₹150K
View Report (Overdue EMI...
As of Sep 24, 2025, 9:07 AM

Outcome

- Reports summarize business data and provide insights.
- Dashboards visualize key loan and repayment metrics for managers.
- Sensitive fields are protected and security issues reviewed, ensuring data privacy.

Phase 10 : Quality Assurance Testing

Objective

The goal of this phase is to validate every feature built in FinSmart CRM (Phases 1–9) by preparing and executing test cases. Testing ensures the system meets business requirements, functions correctly, and is user-friendly.

Test Cases

Test Case 1 — Loan Application Record Creation

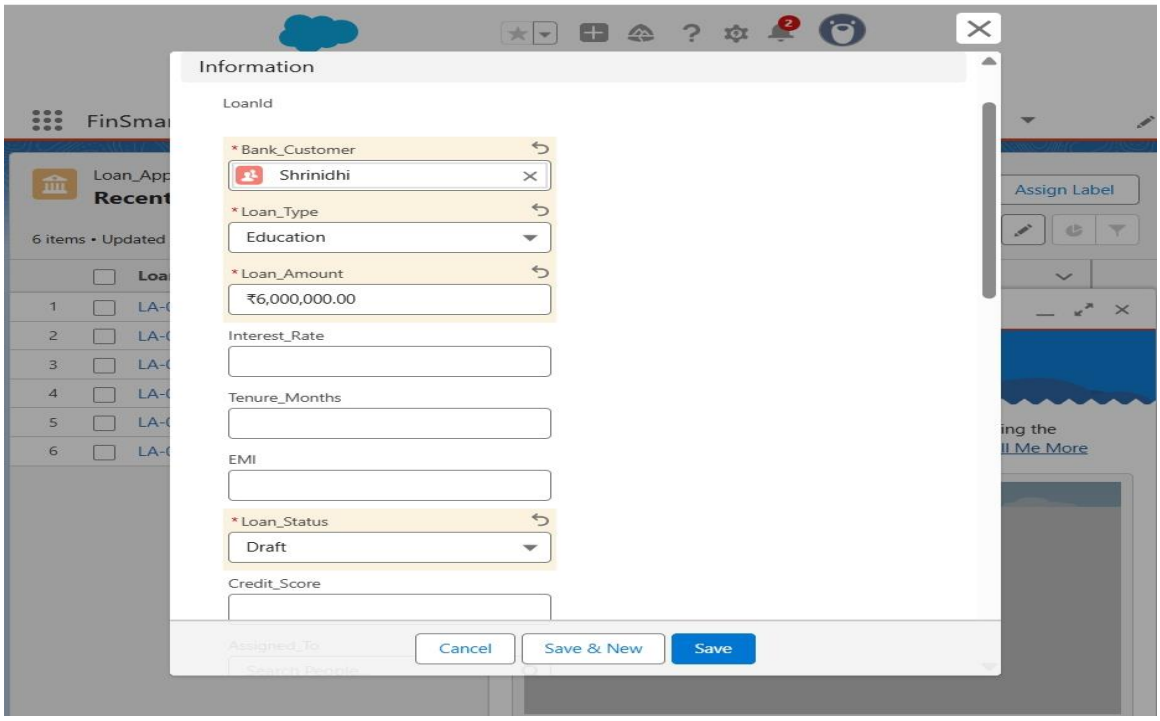
Use Case / Scenario: Loan Application Record Creation

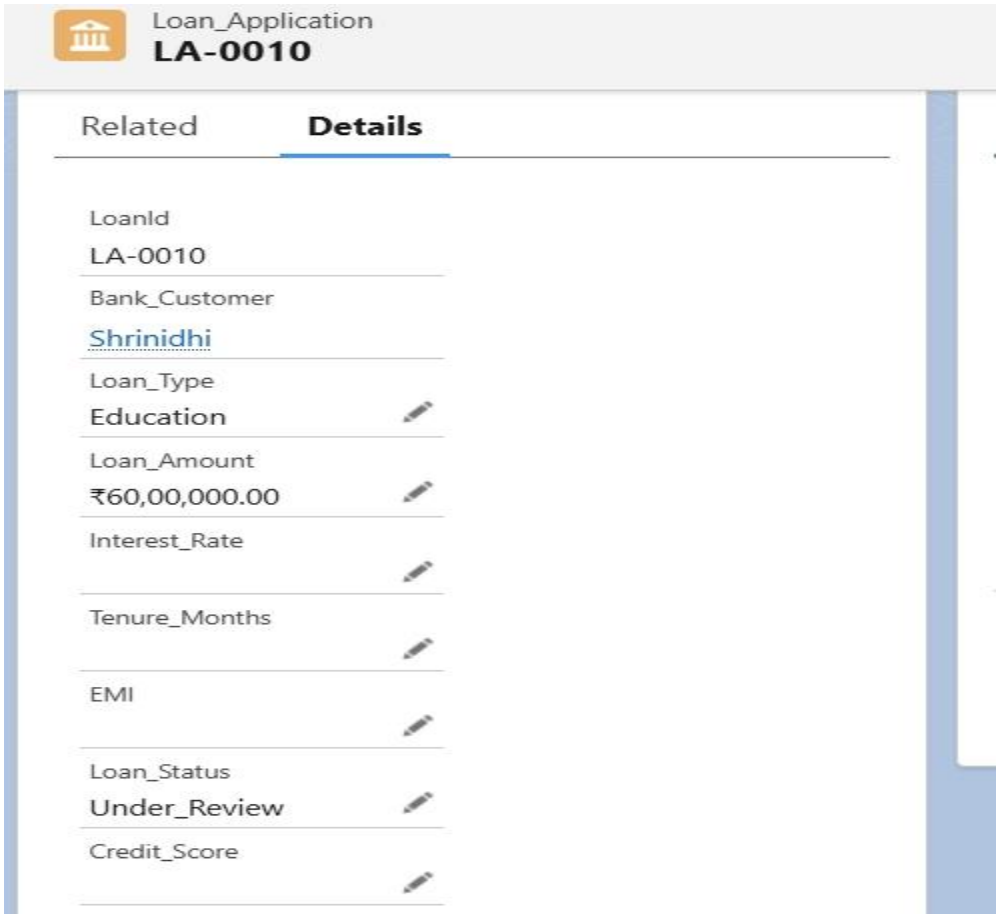
Test Steps (Input):

- Navigate to Loan Applications tab.
- Create new Loan Application with Loan Amount = 600,000.

Expected Result: Loan_Status__c should auto-set to Under_Review (trigger logic).

Actual Result: Loan saved successfully, Loan_Status__c = Under_Review.





Test Case 2 — Validation Rule on Loan Amount

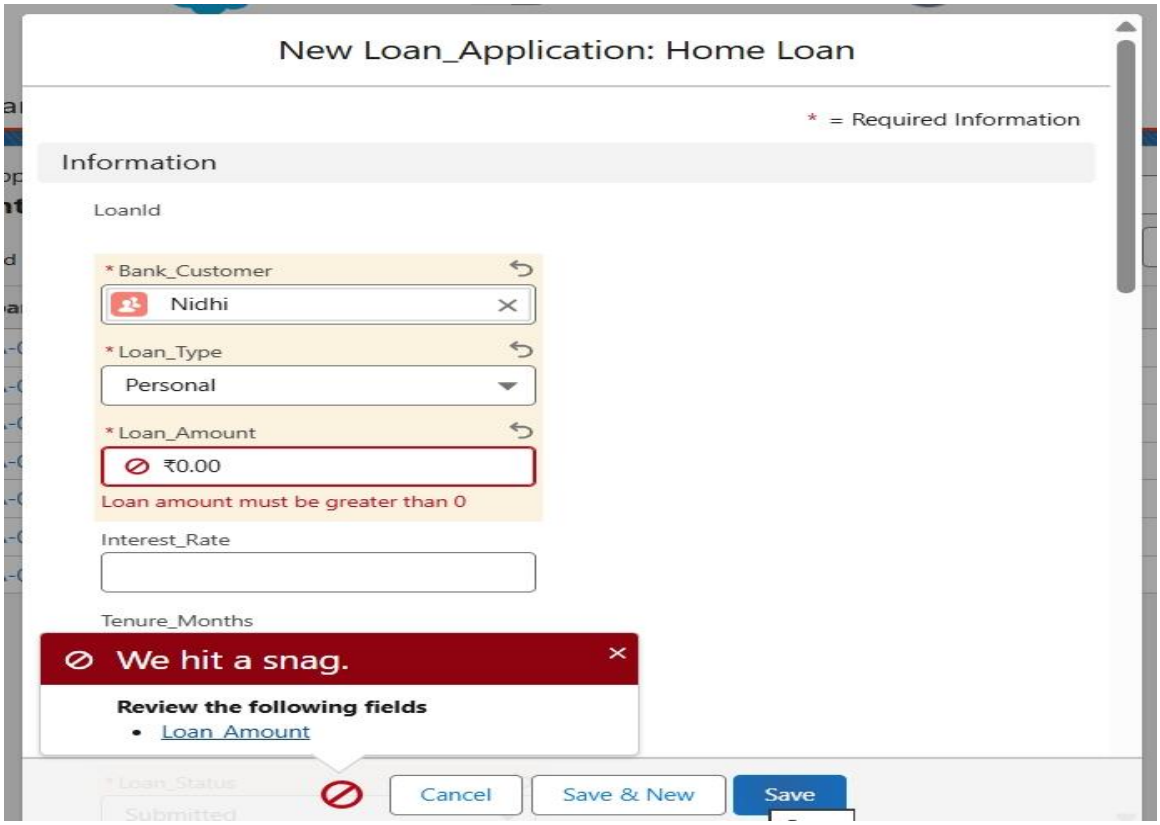
Use Case / Scenario: Validation Rule on Loan Amount

Test Steps (Input):

- Create new Loan Application with Loan Amount = 0.

Expected Result: Error message → 'Loan amount must be greater than 0.'

Actual Result: Validation rule fired correctly. (Screenshot placeholder)



Test Case 3 — Approval Process for High-Value Loan

Use Case / Scenario: Approval Process for High-Value Loan

Test Steps (Input):

- Create Loan Application with Loan Amount = 1,200,000.
- Submit for Approval.

Expected Result: Loan_Status__c → Under_Review. Approval request sent to Manager.

Actual Result: Approval request received. Loan_Status__c updated to Approved.

Test Case 4 — EMI Reminder Scheduled Flow

Use Case / Scenario: EMI Reminder Scheduled Flow

Test Steps (Input):

- Create Repayment record with Due_Date = Today.
- Run scheduled flow.

Expected Result: Customer receives reminder email.

Actual Result: Test email triggered successfully.

Test Case 5 — EMI Calculation (Apex Helper Class) Use Case / Scenario: EMI Calculation (Apex Helper Class)

Test Steps (Input):

- Run LoanHelper.calculateEMI(100000, 12, 12).

Expected Result: EMI > 0.

Actual Result: EMI calculated successfully = 8,885.

Test Case 6 — Credit Bureau Integration
Use Case / Scenario: Credit Bureau Integration

Test Steps (Input):

- Insert Loan Application with status Submitted.
- Run CreditBureauService.getCreditScore(loan.Id).

Expected Result: Credit_Score__c updated (>=650 → Under_Review, else Rejected).

Actual Result: Mock API returned 720. Loan updated to Under_Review.

Test Case 7 — Data Import & Duplicate Rule Use Case / Scenario: Data Import & Duplicate Rule

Test Steps (Input):

- Use Data Import Wizard to import Customer.csv (duplicate Email).

Expected Result: Duplicate Rule blocks insertion and shows warning.

Actual Result: Duplicate error triggered correctly.

Test Case 8 — Reports Use Case / Scenario: Reports

Test Steps (Input):

- Navigate to Reports → Loan Funnel Report.

Expected Result: Loans grouped by Loan_Status__c displayed.

Actual Result: Report generated successfully.

Test Case 9 — Dashboard Use Case / Scenario: Dashboard

Test Steps (Input):

- Navigate to Dashboards → Loan Management Dashboard.

Expected Result: KPI Tiles and charts (Loans by Type, Overdue Trend, etc.) should appear.

Actual Result: Dashboard displayed as designed.

Conclusion

The FinSmart CRM Project has successfully implemented Salesforce features covering data modeling, workflows, validation, automation, integration, reporting, and dashboards.

Final Outcome: FinSmart CRM provides a complete Loan & Customer Management solution with automation, compliance, real-time reporting, and seamless user experience for Bank Officers, Risk Analysts, Managers, and Customers.