# Phase 8: Data Management & Deployment

#### 1. Introduction

### **Explanation:**

Data Management and Deployment ensure that the Medical Inventory Management System remains accurate, scalable, and safely transferable between environments.

This phase covers importing, exporting, deduplicating, backing up, and deploying metadata and data efficiently using Salesforce tools like **Data Loader**, **Change Sets**, **SFDX**, and more.

### **Objectives:**

- Manage and migrate data securely across orgs.
- Automate metadata deployment.
- Prevent duplicate or inconsistent records.
- Maintain data integrity and backup.

# 2. Data Import Wizard

#### **Use Case:**

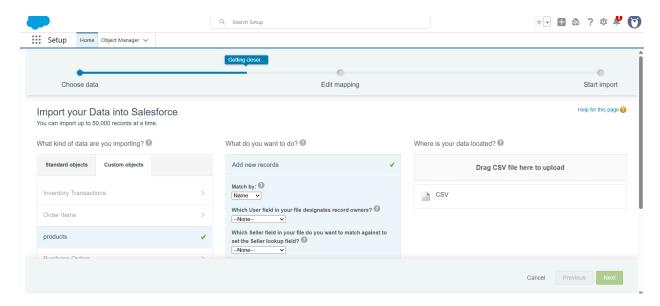
To bulk import Product and Supplier data from CSV files into Salesforce without writing code.

### Scenario:

You receive an Excel sheet from a procurement manager containing a list of 50 medical products and suppliers. You want to load this into Salesforce.

- 1. Navigate to **Setup** → **Data Import Wizard**.
- 2. Choose Object: Product\_\_c.

- 3. Upload CSV file (e.g., Products.csv).
- 4. Map fields (Product Name  $\rightarrow$  Name, Unit Price  $\rightarrow$  Unit Price c).
- 5. Start Import and review the results.



#### 3. Data Loader

#### **Use Case:**

For handling large data volumes (above 50,000 records) or automating imports/exports through scripts.

#### **Scenario:**

You want to bulk import Inventory Transactions (both Stock In and Stock Out) for testing automation flows.

- 1. Download and install **Salesforce Data Loader**.
- 2. Log in using Salesforce credentials (via OAuth).
- 3. Choose **Insert** → Select Object: Inventory\_Transaction\_\_c.
- 4. Browse CSV file (e.g., Inventory\_Transactions.csv).
- 5. Map fields and start upload.

# 4. Duplicate Rules

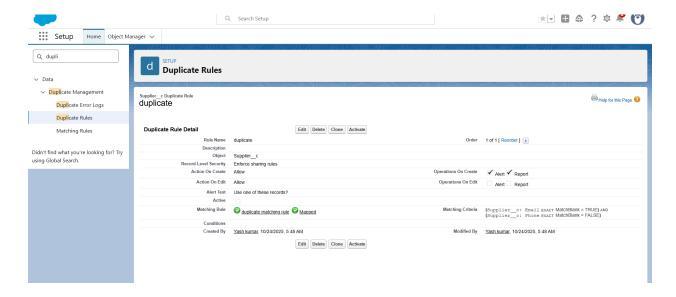
### **Use Case:**

To prevent creation of duplicate Supplier records during import or manual entry.

### **Scenario:**

You want to ensure that no two suppliers with the same email or phone number exist.

- 1. Go to Setup  $\rightarrow$  Duplicate Rules  $\rightarrow$  New Rule.
- 2. Object: Supplier c.
- 3. Matching Rule:
  - o Match on Email\_c OR Phone\_c.
- 4. Action: Block creation or alert user.



### 5. Data Export & Backup

#### **Use Case:**

Regularly back up Salesforce data for disaster recovery and compliance.

### **Steps Implemented:**

- 1. Navigate to Setup  $\rightarrow$  Data Export  $\rightarrow$  Schedule Export.
- 2. Select all custom objects (Product, Supplier, Purchase Order, etc.).
- 3. Choose frequency: Weekly.
- 4. Enable "Include images, documents, attachments".
- 5. Salesforce emails a ZIP download link after export completion.

### 6. Change Sets

#### **Use Case:**

To migrate metadata (objects, fields, flows, triggers, dashboards, etc.) from Developer Org to Production Org securely.

### Scenario:

After final testing, deploy all components of Medical Inventory Management to production.

- 1. Go to Setup  $\rightarrow$  Outbound Change Sets  $\rightarrow$  New.
- 2. Add:
  - o Custom Objects, Fields, Validation Rules
  - o Flows, Apex Classes, Triggers
  - Reports and Dashboards
- 3. Upload to Production  $\rightarrow$  Inbound Change Sets  $\rightarrow$  Deploy.

### 7. Unmanaged vs Managed Packages

#### **Use Case:**

Understanding packaging models helps share or deploy components efficiently.

Type	Description	<b>Use Case</b>
Managed Package	Controlled and versioned; often for AppExchange.	Use for commercial app distribution.
Unmanaged Package	Open and editable; used for internal deployments.	Use for this academic project.

### Scenario:

Created an **Unmanaged Package** to bundle all custom objects, flows, and Apex classes for sharing with teammates.

# **Steps Implemented:**

- 1. Setup  $\rightarrow$  Package Manager  $\rightarrow$  New Package.
- 2. Add all relevant components.
- 3. Generate installation link.

# 8. ANT Migration Tool

#### **Use Case:**

Automating metadata deployment through scripts — ideal for large or frequent deployments.

#### Scenario:

Automate transfer of Apex classes and triggers between sandbox and production.

### **Steps Implemented:**

- 1. Install **Apache ANT** and configure Salesforce credentials in build.properties.
- 2. Create package.xml listing metadata components.

### 9. VS Code & SFDX

#### **Use Case:**

Salesforce DX (Developer Experience) is used for modern development and version control.

#### **Scenario:**

Develop Apex triggers and push them to Salesforce Org using Visual Studio Code.

### **Steps Implemented:**

- 1. Install Salesforce CLI (SFDX).
- 2. In VS Code:

sfdx force:auth:web:login -a DevHub sfdx force:org:create -f project-scratch-def.json -a InventoryScratch

3. Edit triggers/classes directly in VS Code.

