**Phase 3 — Data Modeling & Relationships**

**Goal:** design a clean, secure, scalable data model in Salesforce that supports donation processing, campaign tracking, volunteer management and AI-driven insights.

**1) Standard & Custom Objects**

* **Standard objects** are Salesforce-built (Account, Contact, Campaign, Opportunity, etc.). Use them where they fit — e.g., Contact for a donor or volunteer, Campaign for fundraising drives.
* **Custom objects** (<Name>\_\_c) are created to hold project-specific data (e.g., Donation\_\_c, Program\_\_c, Impact\_Report\_\_c).

**How this maps to the NGO system**

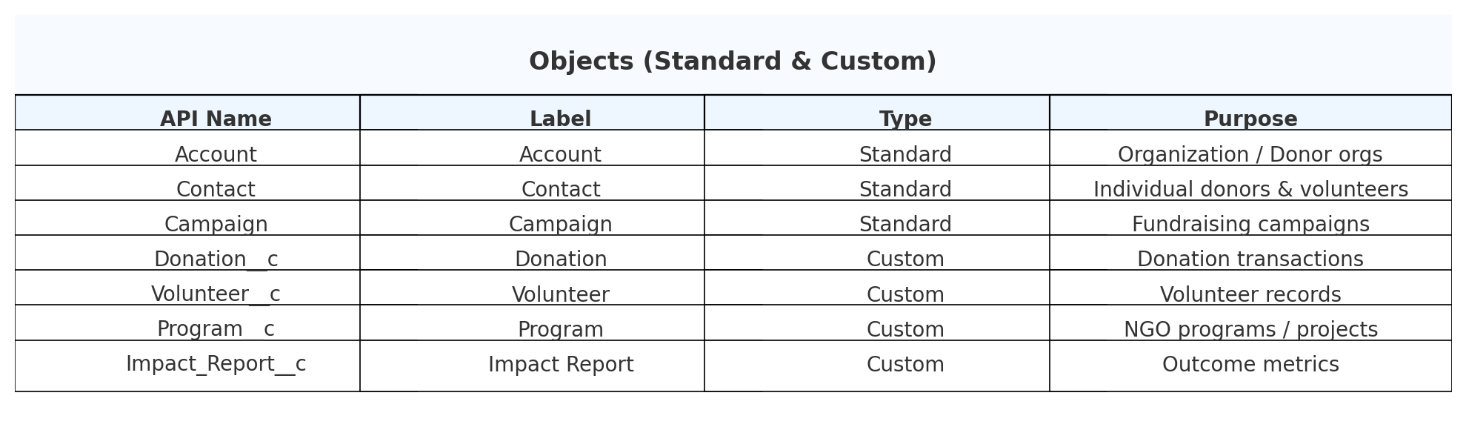
Typical object list:

* **Standard:** Contact (donors/volunteers), Account (donor organizations), Campaign (fundraising campaigns).
* **Custom:**
  + Donation\_\_c — each transaction (amount, date, payment method, receipt status).
  + Program\_\_c — programs/projects that receive funding.
  + Impact\_Report\_\_c — outcome metrics and reports (beneficiaries served, KPIs).
  + Volunteer\_\_c — volunteer detail if you need a richer volunteer record than Contact.

**How to View Standard Objects**

1. **Login** to Salesforce Developer Org.
2. Click on the ⚙️ **Setup** icon (top-right).
3. In the left sidebar Quick Find box → type **Object Manager**.
4. You’ll see a list of all objects.
   * Scroll → you’ll find **Account, Contact, Campaign, Opportunity, Case** etc.

This confirms standard objects are already available.



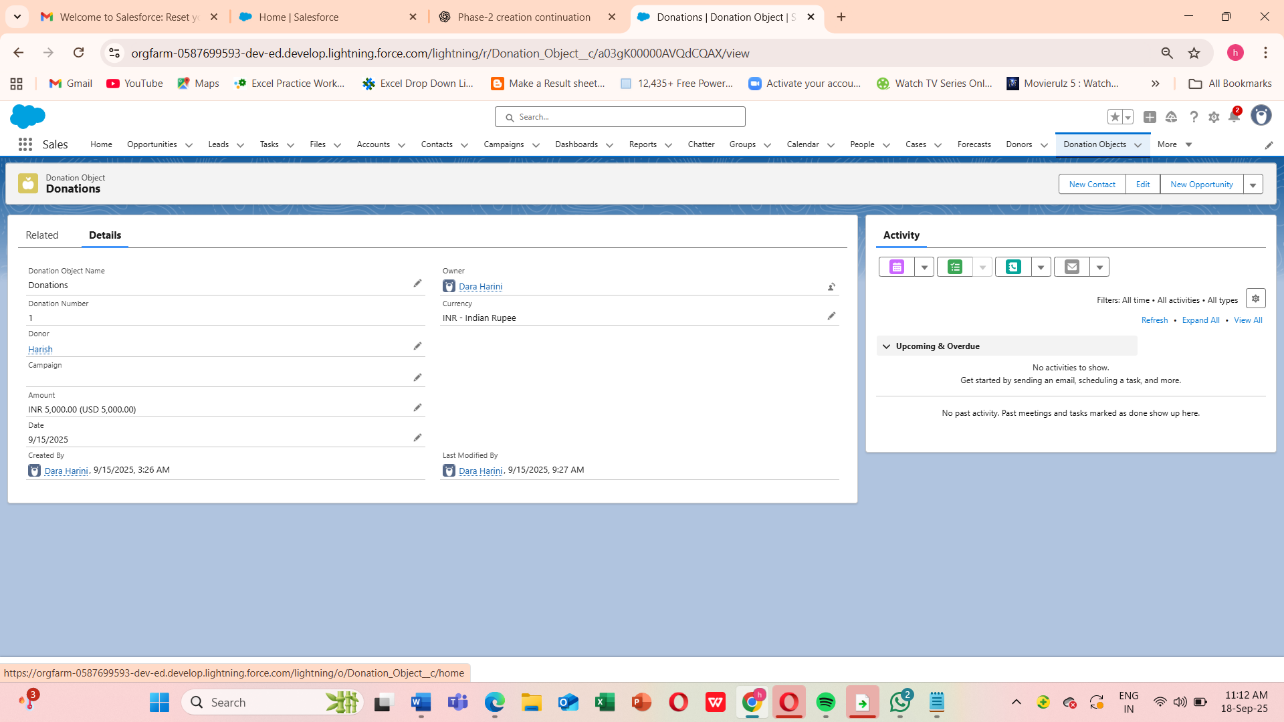
**How to create a custom object (step-by-step)**

1. Setup → **Object Manager** → **Create** → **Custom Object**.
2. Fill fields: **Label**, **Object Name (API)**, **Record Name** (Text or Auto-Number), enable **Allow Reports**, **Allow Activities**, enable **Track Field History** as needed.
3. Save and set permissions (Profiles / Permission Sets).

**Verify Your Custom Object**

After saving:

* Salesforce redirects to **Object Manager → Donation**.
* Here you can add **Fields**, **Page Layouts**, **Record Types**, etc.
* In the **App Launcher (9 dots)** → search “Donations” → you’ll see the new tab created.

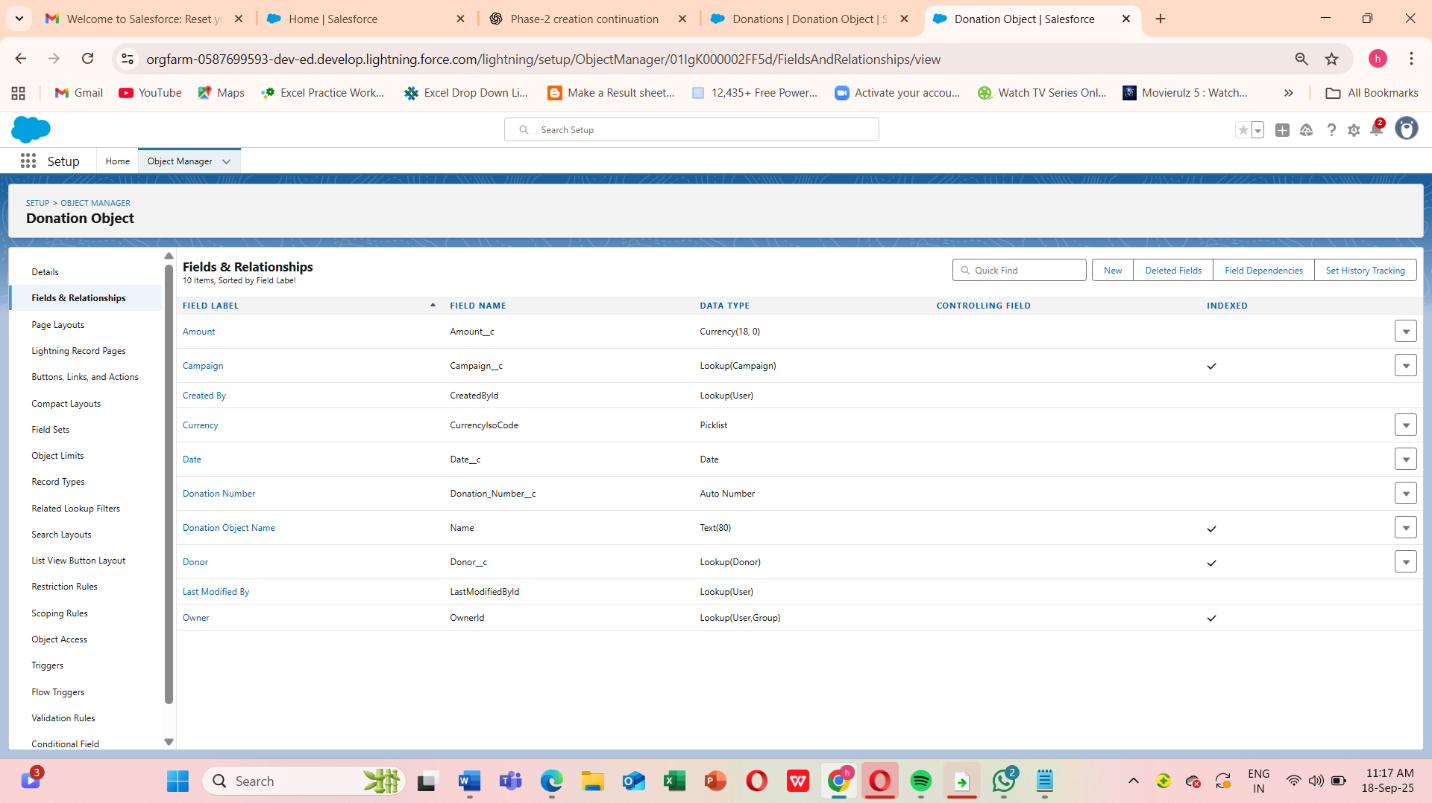




**2) Fields**

**Field types you will use (recommended for Donation)**

* Donation\_Amount\_\_c — **Currency** (required).
* Donor\_\_c — **Lookup(Contact)** (required).
* Donation\_Date\_\_c — **Date** (required).
* Payment\_Method\_\_c — **Picklist** (e.g., Credit Card, Bank Transfer, Cash).
* Is\_Recurring\_\_c — **Checkbox**.
* Receipt\_Sent\_\_c — **Checkbox**.
* Campaign\_\_c — **Lookup(Campaign)**.



**Creating a field**

1. Setup → Object Manager → Select object (Donation) → **Fields & Relationships** → **New**.
2. Choose field type, click Next, configure label, length, help text, and set **Required** if needed.
3. Set **Field-Level Security** (Profiles) and add to Page Layouts. Save.

**Important field considerations**

* **Required vs optional:** Only make required what’s essential (e.g., donor, amount, date).
* **Help text**: Add help text to guide users.
* **Indexing:** Standard fields and some custom fields are indexed automatically. If queries get slow, consider marking a field as External ID or creating a custom index (via Salesforce support) for very large datasets.
* **Unique/External ID:** For import/lookup/upsert flows, mark donor external ID (like DonorID\_\_c) to simplify deduping.

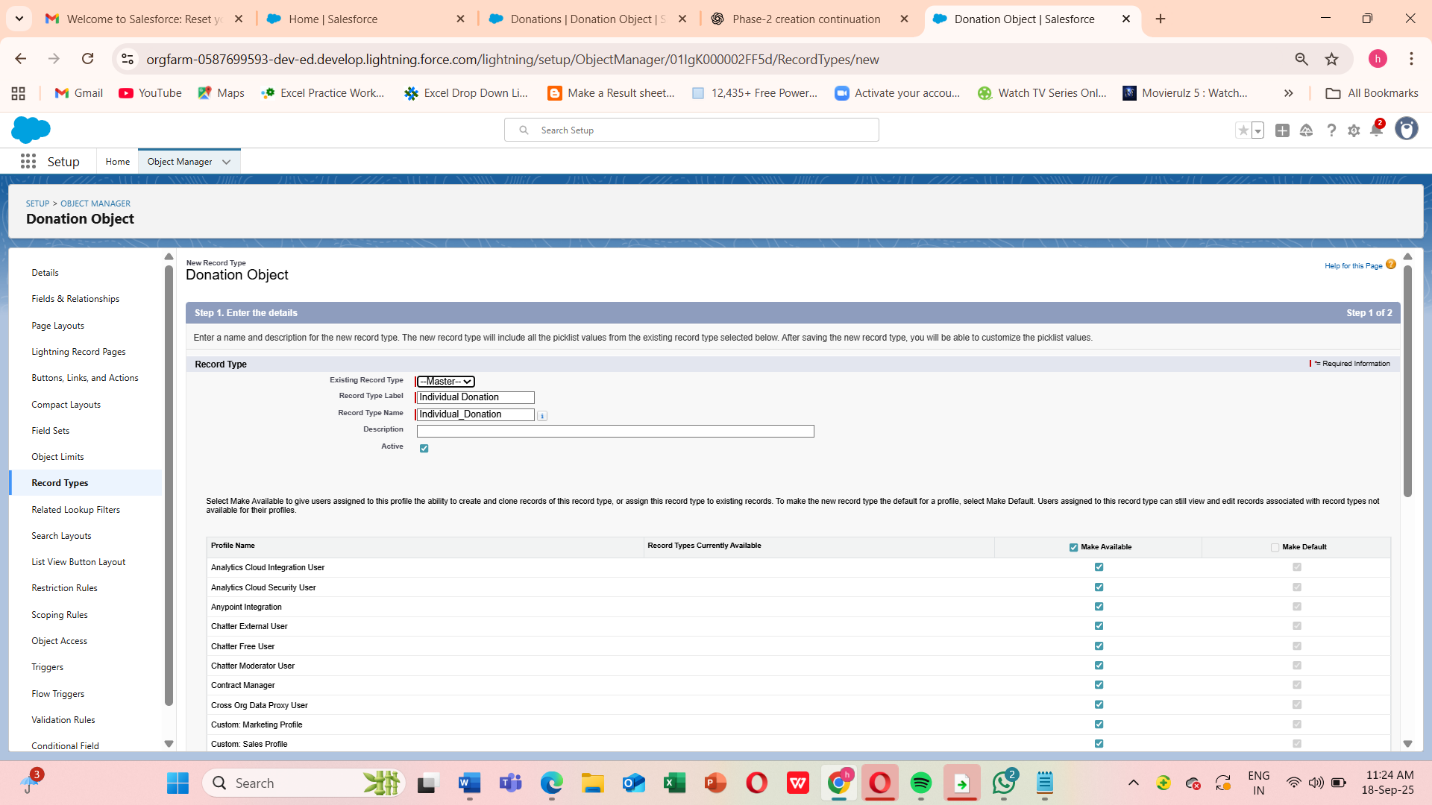
**3) Record Types**

**What record types do**

Record Types let you support different business processes, picklist values, and page layouts on the same object.

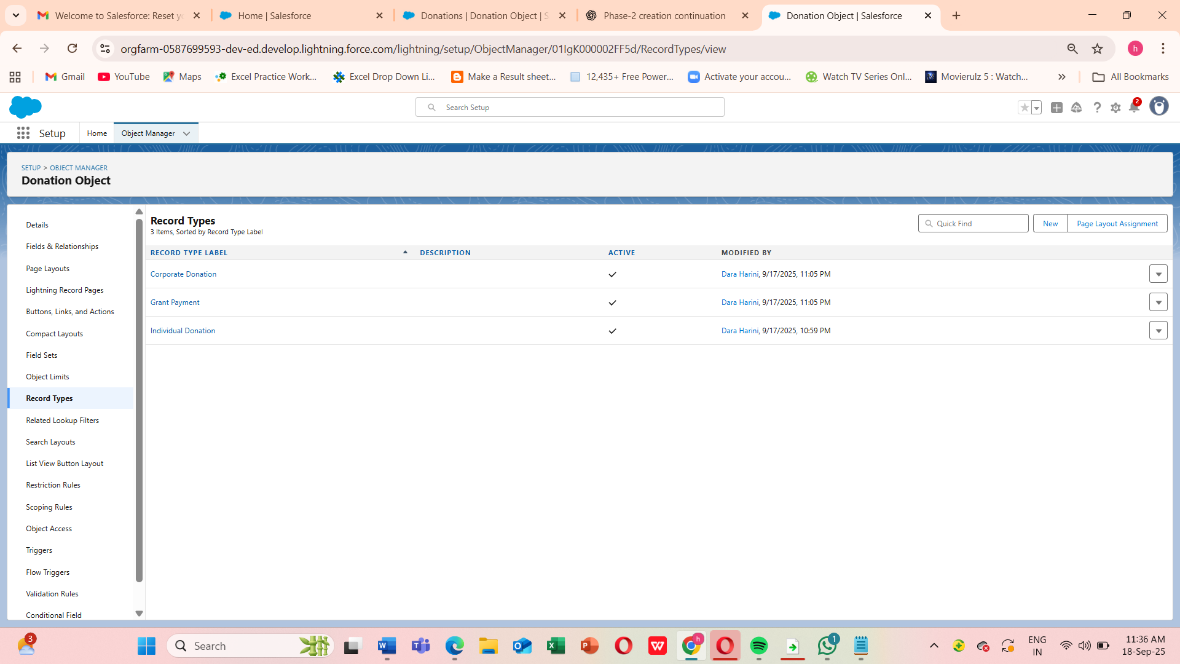
**Example use for donations**

* **Individual Donation** — fields relevant to an individual donor (Contact lookup prominent).
* **Corporate Donation** — additional fields for Company\_Name\_\_c, CSR\_Project\_\_c, Tax\_Exemption\_\_c.
* **Grant Payment** — additional compliance fields (Grant ID, Reporting requirements).



**Create Record Type (steps)**

1. Setup → Object Manager → Donation → **Record Types** → **New Record Type**.
2. Name the record type, set the default picklist values specific to that type.
3. Assign the record type to profiles, then create and assign matching Page Layouts to each profile/record type combo.



**4) Page Layouts**

**Purpose**

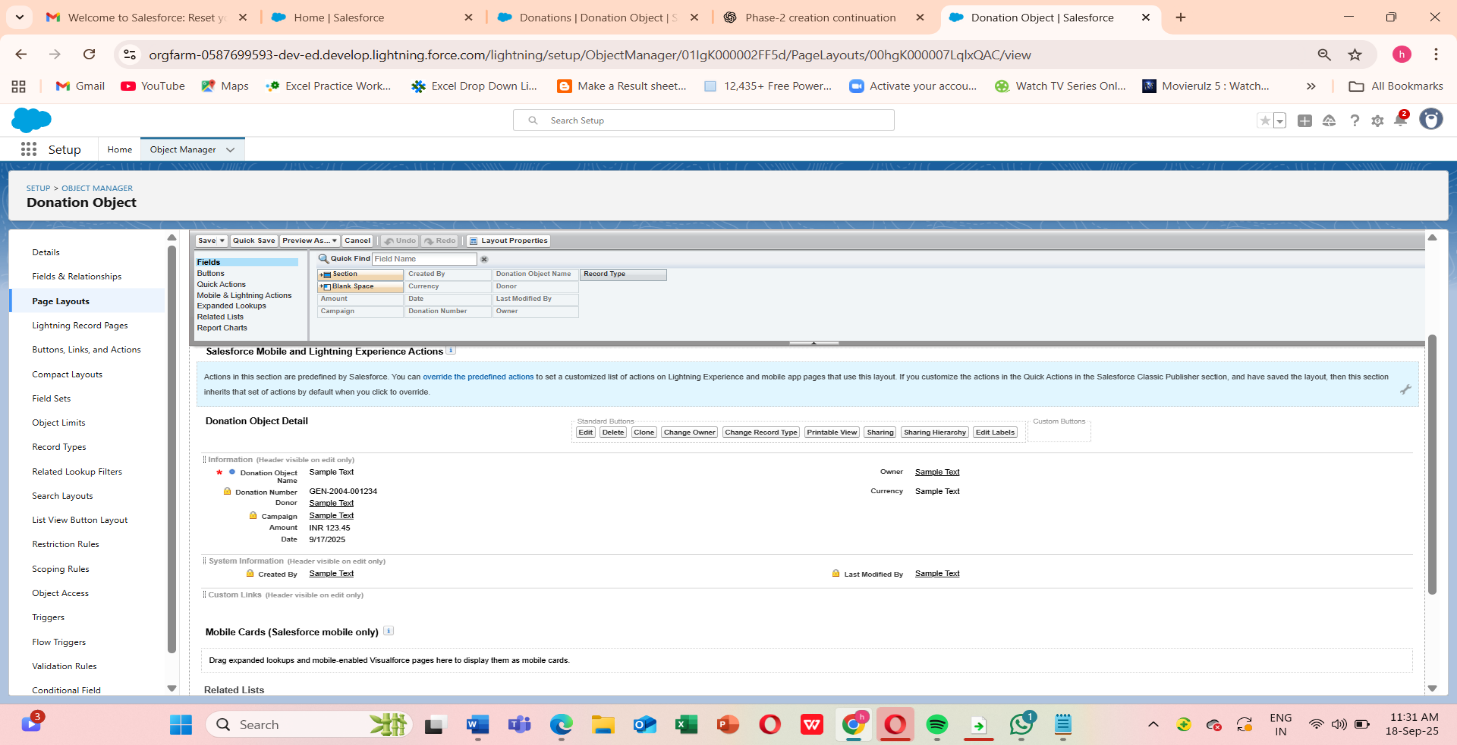
Page layouts control field order, sections, related lists and which quick actions appear on the record page.

**NGO examples**

* **Donation – Individual Layout**: top section with Donor (Contact), Amount, Date; a second section for Receipt and Payment Method; related list shows Impact Allocations (if donation is allocated to multiple programs).
* **Donation – Corporate Layout**: includes Company contact, CSR approvals, Tax fields.

**Edit Page Layout (steps)**

1. Setup → Object Manager → Donation → **Page Layouts** → Edit layout.
2. Drag fields into sections, configure related lists (e.g., Donation Allocations, Notes & Attachments).
3. Save and **Preview**.



**5) Compact Layouts**

**What they are**

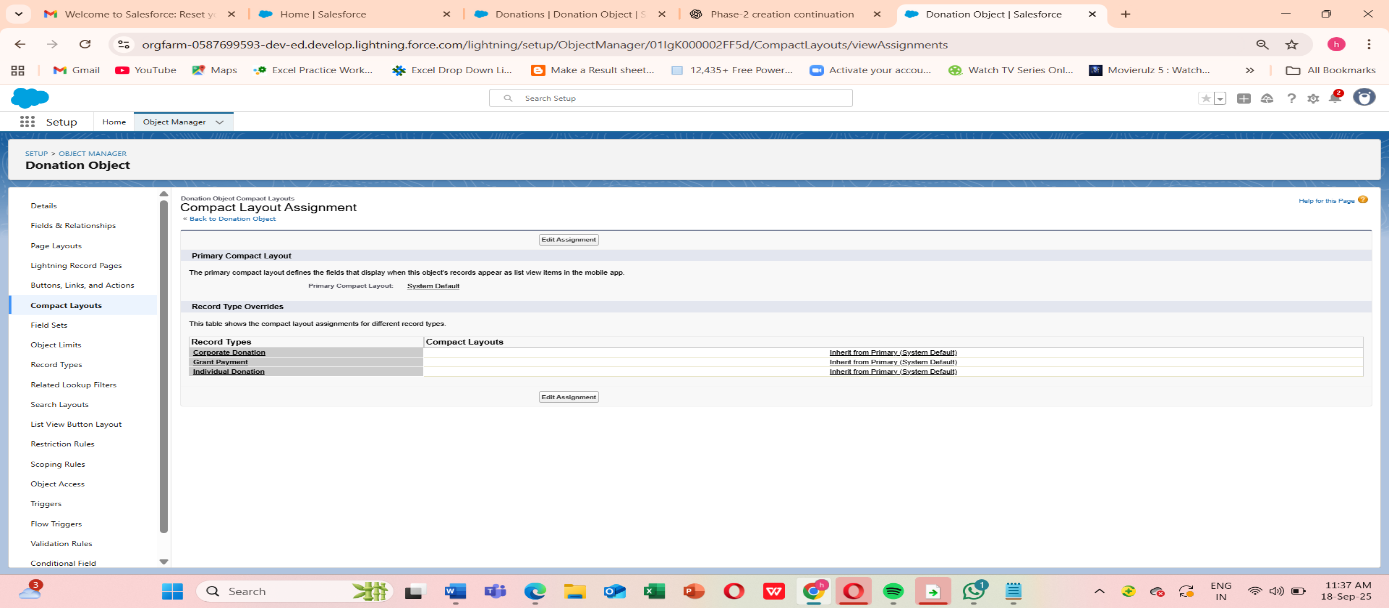
Compact layouts control which fields appear in the Highlights panel (desktop) and in mobile record lists.

**Recommended compact layout for Donation**

* Donor | Donation Amount | Donation Date | Receipt Sent (Y/N)

**How to create**

1. Setup → Object Manager → Donation → **Compact Layouts** → **New**.
2. Add fields, save → Assign the compact layout to the Record Type or Org.



Compact layouts give staff a fast summary in list views and mobile, which is very useful for fundraisers on the go.

**6) Schema Builder in Salesforce**

**1. What is Schema Builder?**

* A **drag-and-drop UI tool** in Salesforce to visualize and manage objects.
* Shows:
  + Standard & Custom objects
  + Fields
  + Relationships between objects
* Helps NGOs see **how Donors, Donations, Campaigns, and Programs connect**.

**2. Steps to Open Schema Builder**

1. **Login** to Salesforce Developer Org.
2. Click ⚙️ **Setup** (top right).
3. In **Quick Find box** → type **Schema Builder**.
4. Select **Schema Builder** → a canvas opens with object diagrams.

**3. Configure Schema Builder**

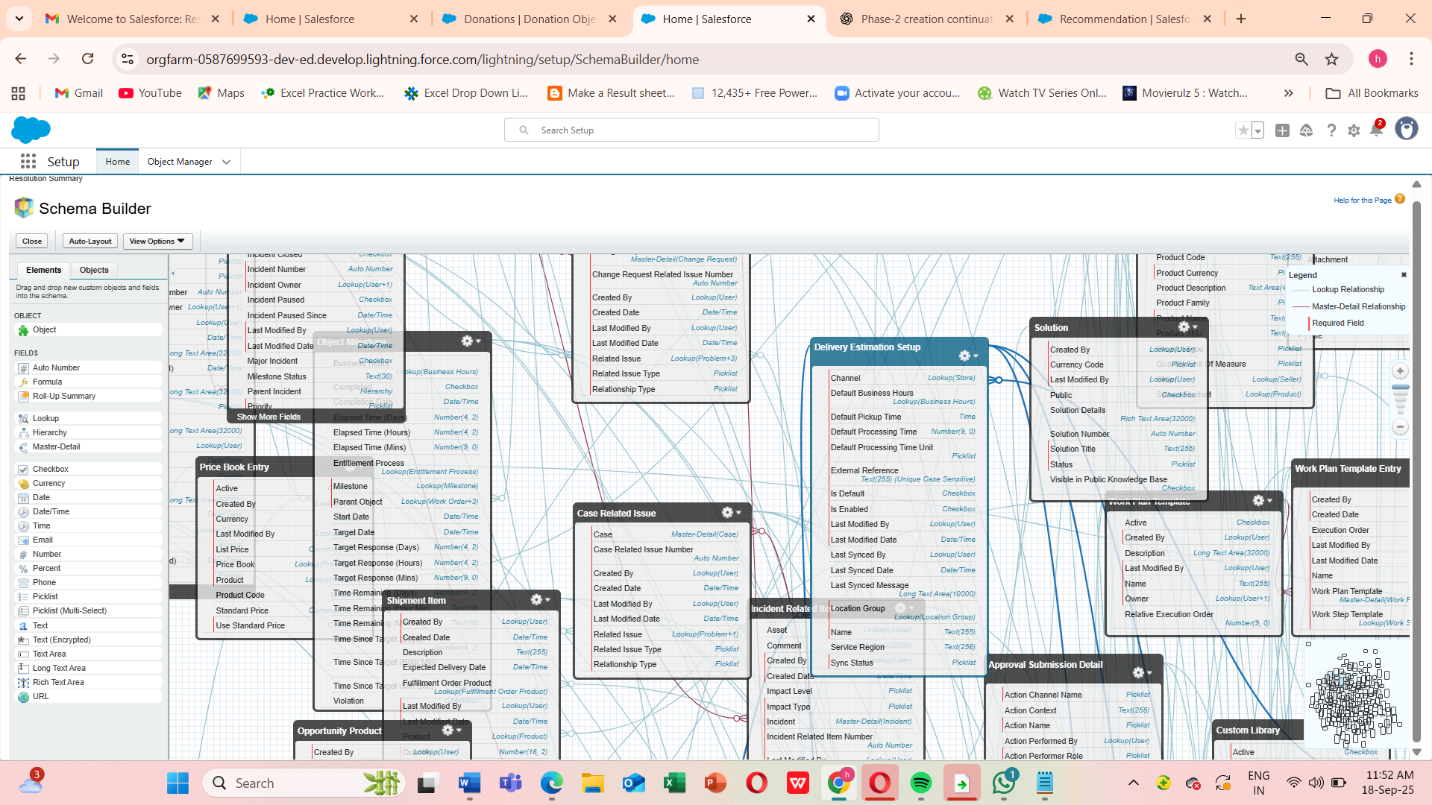
* On the **left panel**, you’ll see:
  + **Elements** (Fields: Lookup, Formula, etc.)
  + **Objects** list (Account, Contact, Donation\_\_c, Program\_\_c, Impact\_Report\_\_c, Volunteer\_\_c).
* Check the boxes for:
  + **Contact**
  + **Campaign**
  + **Donation\_\_c**
  + **Program\_\_c**
  + **Impact\_Report\_\_c**

This will display only our NGO-related objects.

**4. Relationships You’ll See**

Once loaded, Salesforce automatically shows relationship lines between objects:

* **Contact → Donation\_\_c**
  + Lookup Relationship (a Donation belongs to a Donor).
* **Campaign → Donation\_\_c**
  + Lookup Relationship (a Donation links to a Campaign).
* **Program\_\_c → Impact\_Report\_\_c**
  + Master-Detail Relationship (each Report is tied to one Program).
* **Donation\_\_c ↔ Program\_\_c** (via **Junction Object** Donation\_Allocation\_\_c)
  + Many-to-Many relationship between Donations & Programs.



**7) Lookup vs Master-Detail vs Hierarchical Relationships**

This is crucial — pick the right relationship type for the business need.

* **Lookup**
  + Ownership: **Independent** — the child record does not inherit parent’s owner/sharing.
  + Roll-up Summary: **No**.
  + Required: Optional or required depending on your choice when creating the field.
  + Cascade Delete: Optional (configurable for lookups: standard behavior is not to delete child automatically).
  + Use when: you need a flexible relationship, or both objects must exist independently (e.g., Campaign lookup on Donation).
* **Master-Detail**
  + Ownership: **Child inherits parent owner**; sharing controlled by parent.
  + Roll-up Summary: **Yes** (parent can have roll-ups like Total Donations).
  + Required: Yes (child must have a parent).
  + Cascade Delete: Yes (deleting parent deletes child).
  + Use when: the child should not exist without the parent and you want roll-up summaries (e.g., Donation\_Allocation\_\_c as child of Donation and Program when built as junction with two master-detail fields).
* **Hierarchical**
  + Special relationship type available only between User and User (e.g., Manager → Employee).
  + Use for user-to-user reporting structure only.

**Practical NGO examples**

* **Lookup**: Donation\_\_c.Campaign\_\_c = Lookup(Campaign) — Campaigns can exist separately from donations.
* **Master-Detail**: If you want Program\_\_c to own Impact\_Report\_\_c and roll up totals of donations allocated, use master-detail.
* **Hierarchical**: Use for User.ManagerId relationships in org chart.

**How to create a relationship field**

1. Setup → Object Manager → child object → **Fields & Relationships** → **New**.
2. Choose **Lookup Relationship** or **Master-Detail Relationship**, follow wizard and select parent object.
3. Configure sharing behavior and delete cascade (for lookup, optional settings may be offered).

**Important notes**

* You can convert a **Lookup → Master-Detail** only if no orphaned child records exist and the child has a value for parent for all records.
* Use **Master-Detail** if you need roll-up summaries; otherwise use lookup to preserve independence and flexibility.

**8) Junction Objects (Many-to-Many)**

What is a junction object

Junction object is a custom object with two master-detail relationships to model many-to-many relationships.

NGO example

* Scenario: A single donation may be split across multiple Programs (Donation allocations). A Program may receive funds from many donations.
* Solution: Create a junction object Donation\_Allocation\_\_c with:
  + Donation\_\_c — Master-Detail → Donation
  + Program\_\_c — Master-Detail → Program
  + Additional fields: Allocation\_Amount\_\_c (Currency), Allocation\_Percentage\_\_c (Formula or number), Notes\_\_c.

**Steps to Create External Objects in Salesforce**

1. Go to **Setup → Quick Find → External Data Sources**.
2. Click **New External Data Source**.
   * **Label:** Payment Gateway
   * **Name:** Payment\_Gateway
   * **Type:** Salesforce Connect: OData 4.0
   * **URL:** (URL from payment gateway API, e.g., https://api.razorpay.com/odata/v1)
   * Save.
3. After saving → Click **Validate and Sync**.
   * Salesforce shows a list of **external tables**.
   * Select the one you need, e.g., **Transactions**.
   * Salesforce creates an **External Object: Payment\_Transaction\_\_x**.
4. Go to **Object Manager → Payment\_Transaction\_\_x**.
   * Add fields/relationships if needed.
   * Example: Lookup → Donation\_\_c

**4. Verifying External Object**

* Go to **App Launcher → Search Payment Transactions**.
* You will see a new tab.
* When opened → Salesforce fetches **live transaction data** from the external system.

**9) External Objects (Salesforce Connect)**

External Objects let Salesforce show data that lives outside Salesforce (OData, external DBs) without copying it into your org. Records appear with suffix \_\_x.

**Typical NGO use cases**

* Real-time view of payment gateway transactions (Payment\_Transactions\_\_x) for reconciliation without importing full transaction history.
* Integration with a legacy donor database for reference data.

**How to set up (overview)**

1. Setup → **External Data Sources** → **New**.
   * Choose type: **OData 2.0/4.0**, **Salesforce Connect: OData 2.0**, or other connectors.
   * Fill in endpoint URL, authentication (Named Credential recommended).
2. Click **Validate and Sync** → Salesforce will fetch object tables and create **External Objects**.
3. Create **External Lookup** or regular lookup from Donation\_\_c to the external object to relate transactions to donations.

**Limitations & considerations**

* Some platform features (Apex triggers, standard reports) behave differently with external objects — check feature compatibility.

External objects can affect performance depending on external provider speed. For heavy reporting you may want periodic replication into Salesforce (ETL) or use a middleware cache.

* Salesforce Connect may require licensing (often paid add-on).

**Additional data modeling best practices (NGO & AI context)**

1. **Design for analytics**
   * Store donation time series (donation date + amount) and donor attributes (join date, segmentation) so AI models can compute lifetime value, churn risk, and next best action features.
   * Keep historical snapshots or use pipelines to maintain time-based aggregates for model training.
2. **Use meaningful API names and descriptions**
   * Example: Donation\_Amount\_\_c (clear), DonationDate\_\_c not DD — helps developers & integrations.
3. **Validation Rules & Data Quality**
   * Always validate critical fields (amount > 0, donor exists). This improves model quality and reporting.
4. **Field-Level Security & PII**
   * Hide sensitive donor PII using Field-Level Security and Permission Sets. Use encryption for very sensitive fields.
5. **Indexes & Performance**
   * For large datasets, consider selective filters on indexed fields, External ID fields for import/upsert performance, and Big Objects if millions of records are expected.
6. **Naming & Documentation**
   * Keep a data dictionary (object, field, description, example) — very useful for AI features and new team members.
7. **Test Thoroughly**
   * Use your Developer Org (or Sandbox in Enterprise) to test roll-up summaries, record type assignments, sharing & security behavior, and data flows.