**Phase 2: Data Model Design & Custom Objects**

**Objective of this Phase**

The goal of Phase 2 was to **design the Salesforce Data Model** that supports the Loyalty Program, Rewards, and Error Logging.  
We created **custom objects, fields, and relationships** to capture all the data needed for automation and reporting.

1. **Custom Objects Created**

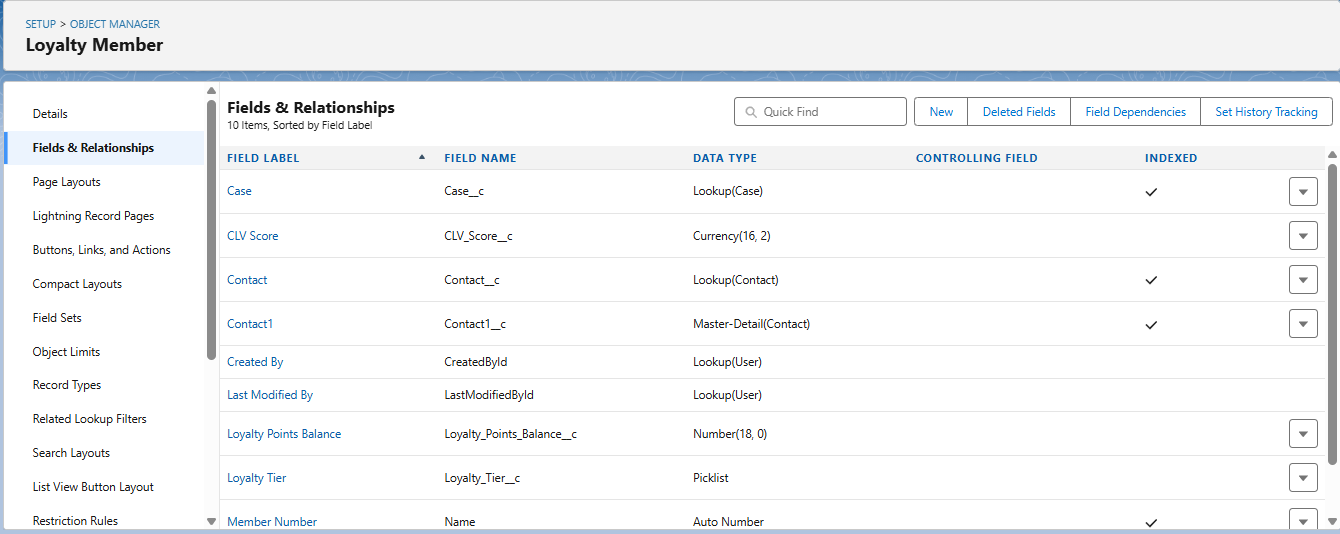
We created the following **custom objects**:

**Loyalty Member (Loyalty\_Member\_\_c)**

* + Represents each customer’s loyalty program profile.
  + Linked to the Contact object.
  + Tracks balance, tier, and related transactions.

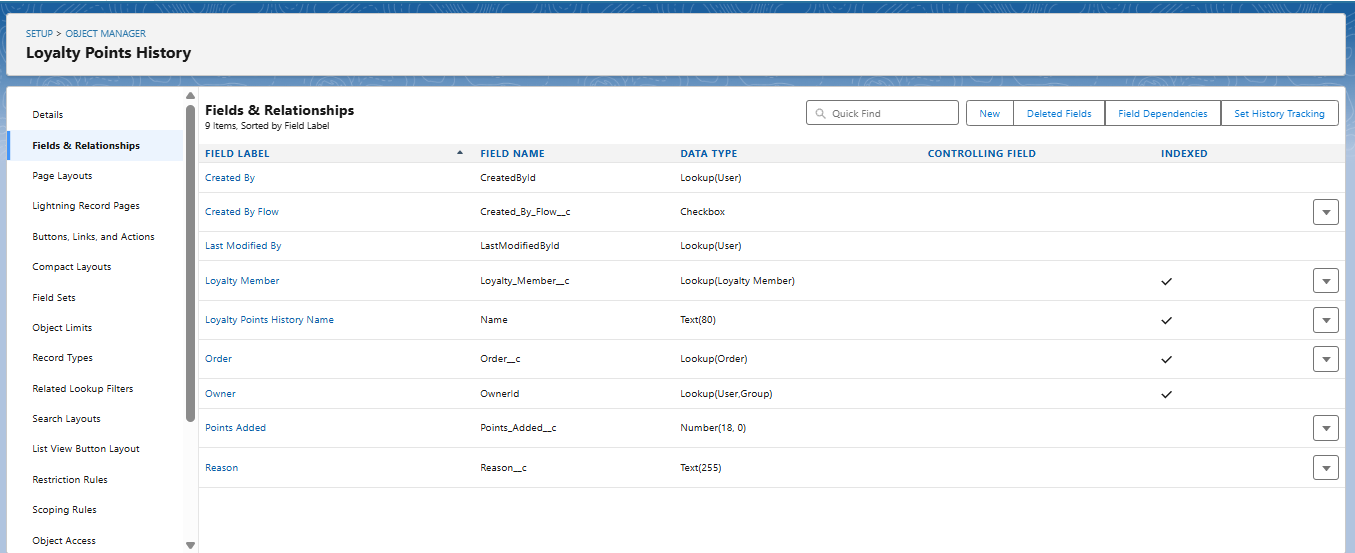
Key Fields:

* + Contact\_\_c → Lookup(Contact) (primary relationship).
  + Loyalty\_Points\_Balance\_\_c → Number(18,0).
  + Loyalty\_Tier\_\_c → Picklist (Silver, Gold, Platinum).
  + Order\_\_c → Lookup(Order).
  + Case\_\_c → Lookup(Case).
  + CLV\_Score\_\_c → Formula (for Customer Lifetime Value).



1. **Loyalty Points History (Loyalty\_Points\_History\_\_c)**
   * Keeps track of how points were earned.
   * Logs the source (Order, Manual, Adjustment).

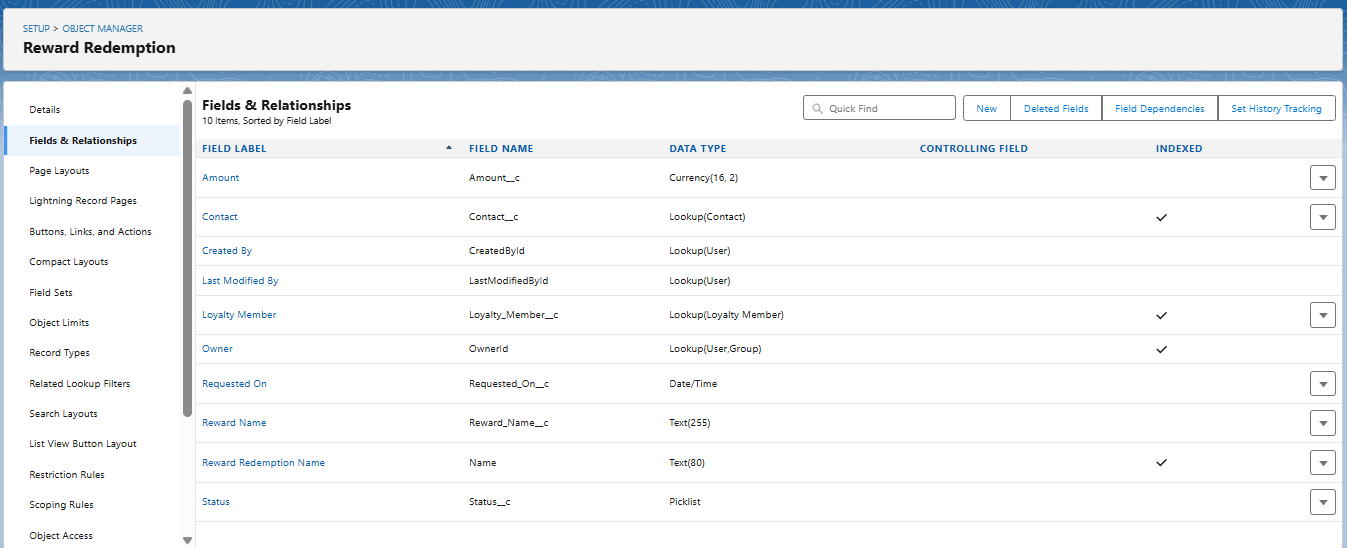
Key Fields:

* + Loyalty\_Member\_\_c → Lookup(Loyalty Member).
  + Order\_\_c → Lookup(Order).
  + Points\_Added\_\_c → Number(18,0).
  + Reason\_\_c → Text (e.g., “Order Activated”).
  + Created\_By\_Flow\_\_c → Checkbox (true if created by automation).

1. **Reward Redemption (Reward\_Redemption\_\_c)**
   * Stores records of rewards redeemed by customers.
   * Can trigger approval if reward exceeds a threshold.

Key Fields:

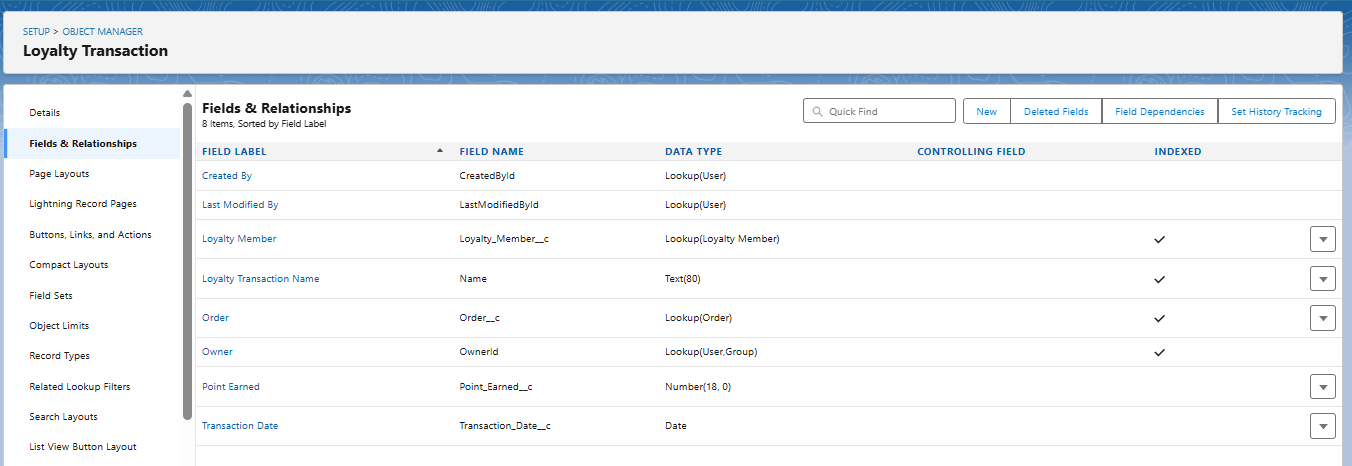
* + Contact\_\_c → Lookup(Contact).
  + Loyalty\_Member\_\_c → Lookup(Loyalty Member).
  + Amount\_\_c → Number (points required).
  + Reward\_Name\_\_c → Text.
  + Requested\_On\_\_c → Date/Time.
  + Status\_\_c → Picklist (Submitted, Approved, Rejected).



1. **Loyalty Transaction (Loyalty\_Transaction\_\_c)** *(Optional reporting object)*
   * Used to store detailed breakdowns of points earned per transaction.
   * Helps with advanced dashboards.

Key Fields:

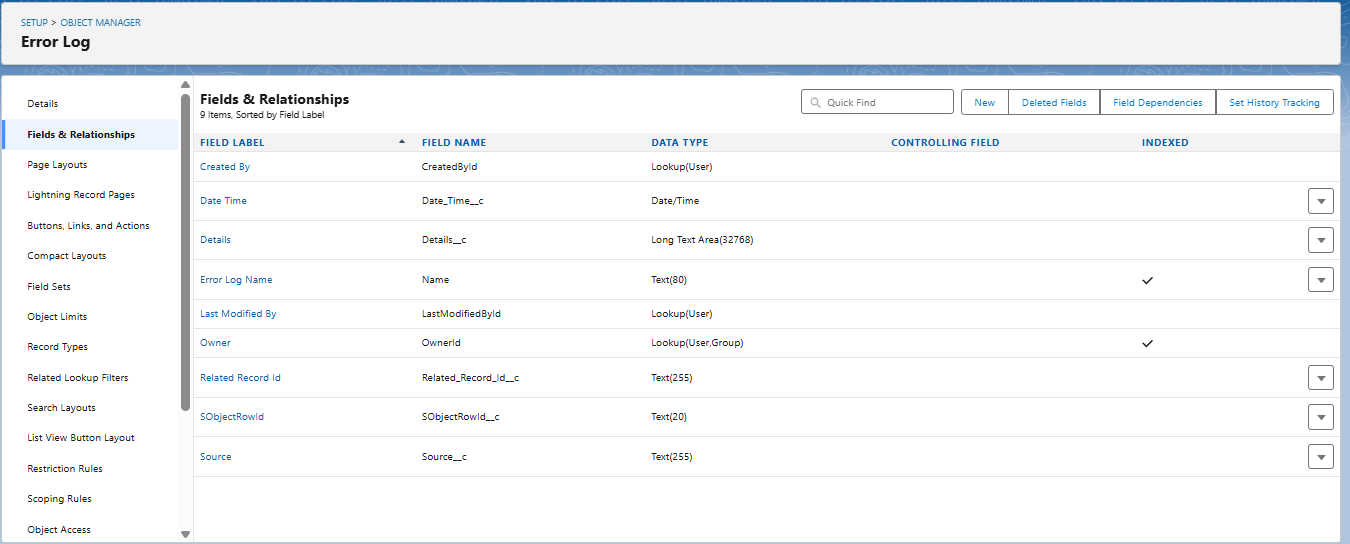
* + Loyalty\_Member\_\_c → Lookup(Loyalty Member).
  + Order\_\_c → Lookup(Order).
  + Point\_Earned\_\_c → Number.
  + Transaction\_Date\_\_c → Date.



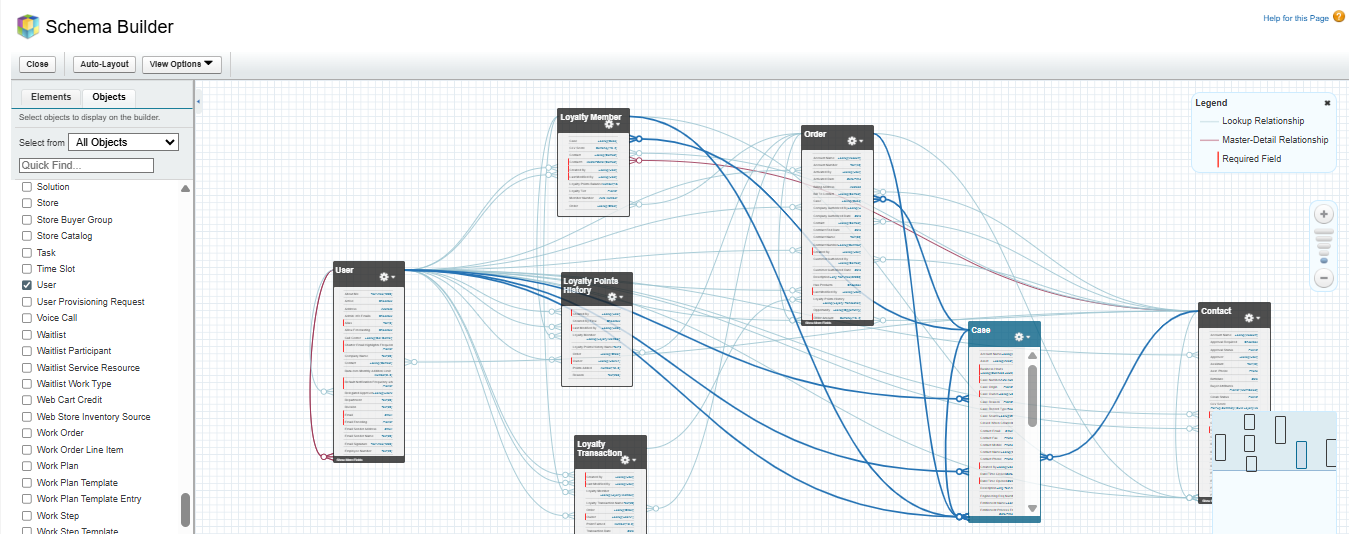
1. **Error Log (Error\_Log\_\_c)**
   * Captures errors from Flows (via fault paths).
   * Allows admins to review what failed and why.

Key Fields:

* + Details\_\_c → Long Text Area (255).
  + Source\_\_c → Text (name of the flow or process).
  + SObjectRowId\_\_c → Text (record id that failed).
  + Related\_Record\_Id\_\_c → Text.
  + Date\_Time\_\_c → Date/Time.



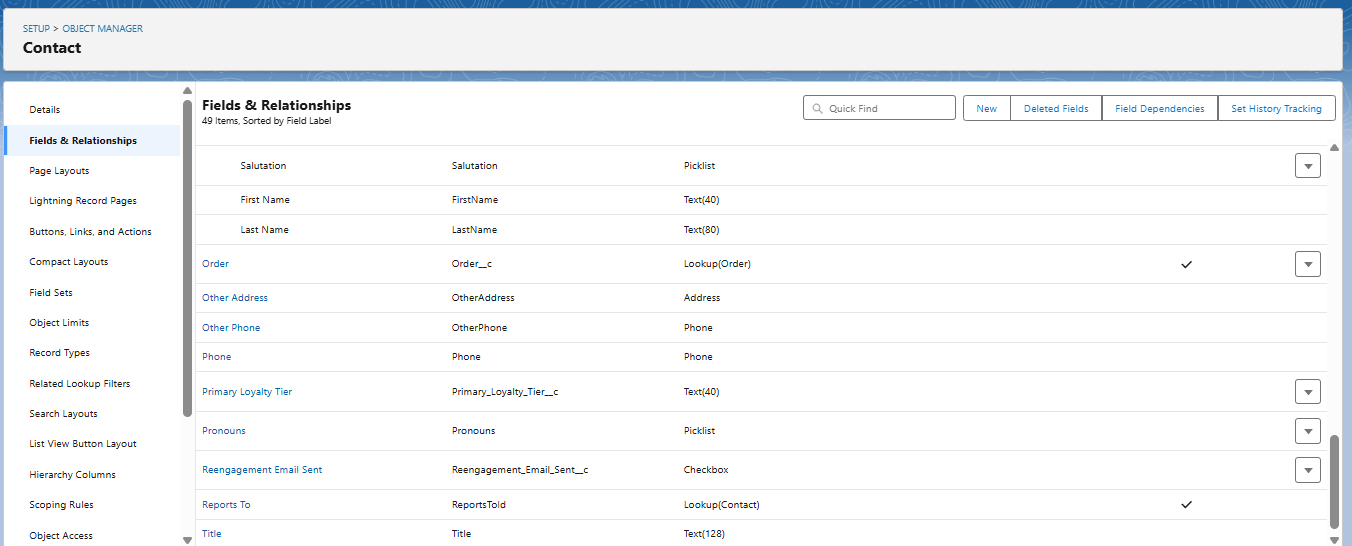
1. **Relationships Between Objects**

* **Contact ↔ Loyalty Member** → One-to-One (each Contact has one Loyalty Member).
* **Loyalty Member ↔ Loyalty Points History** → One-to-Many (points log).
* **Loyalty Member ↔ Reward Redemption** → One-to-Many (customer can redeem multiple rewards).
* **Order ↔ Loyalty Member** → Lookup (links orders to loyalty members).
* **Flows/Triggers ↔ Error Log** → Write errors when something fails

1. **Standard Objects Used**

In addition to custom objects, we also extended **standard objects**:

1. **Contact**
   * Added Primary\_Loyalty\_Tier\_\_c → Formula field to display current tier.
2. **Order**
   * Used Status field (Draft → Activated).
   * Points calculated when **Activated**.
3. **Case**
   * Linked with Loyalty Member for customer service integration.



1. **Validation Rules**

We added **validation rules** to ensure data integrity:

1. **Points Cannot Be Negative** (on Loyalty Member):
2. Loyalty\_Points\_Balance\_\_c < 0

Error: "Points cannot be negative."

1. **Tier Required When Points Exist** (on Loyalty Member):
2. AND(
3. Loyalty\_Points\_Balance\_\_c > 0,
4. ISBLANK(Loyalty\_Tier\_\_c)
5. )

Error: "Tier must be assigned when points exist."

1. **Reward Redemption Must Have Positive Amount** (on Reward Redemption):
2. Amount\_\_c <= 0

Error: "Redemption amount must be greater than zero."

1. **Summary of Phase 2**

At the end of Phase 2, we had:  
✔ All custom objects created.  
✔ Standard objects extended with additional fields.  
✔ Relationships mapped via Schema Builder.  
✔ Validation rules added for data consistency.

This created a **solid foundation** for building Flows, Apex logic, and Approval Processes in later phases.