**Phase 3**

### Standard & Custom Objects:

* **Primary Objective:** To define the complete data model schema by selecting appropriate standard objects and creating all necessary custom entities.
* **Standard Objects Utilized:**
  1. **Accounts:** Used as the base object to manage organizational entities like **Property Owners** and key vendor accounts.
  2. **Contacts:** Used to track individuals, primarily **Tenants**, before their engagement moves to the custom Lease object.
  3. **Reports and Dashboards:** Used for all analytic needs (Phase 9).
* **Custom Objects Created:**
  1. Seven custom objects were created to house the core business logic (e.g., **Property**, **Lease**, **Payment**).
  2. This set of custom objects forms the entire relational schema for the RentEase application, driving all future configuration and automation.

**Standard & Custom Objects: Implementation**

**1. Define Standard Object Usage**

* **Action:** We will utilize the following standard objects:
  + **Accounts:** To track **Property Owners** and any related Business Partners (e.g., vendors).
  + **Contacts:** To track individual people, specifically the **Tenants**, before they are linked to a Lease.
  + **Reports/Dashboards:** For all system analytics.

**2. Create Custom Objects (Manual Creation in New Org)**

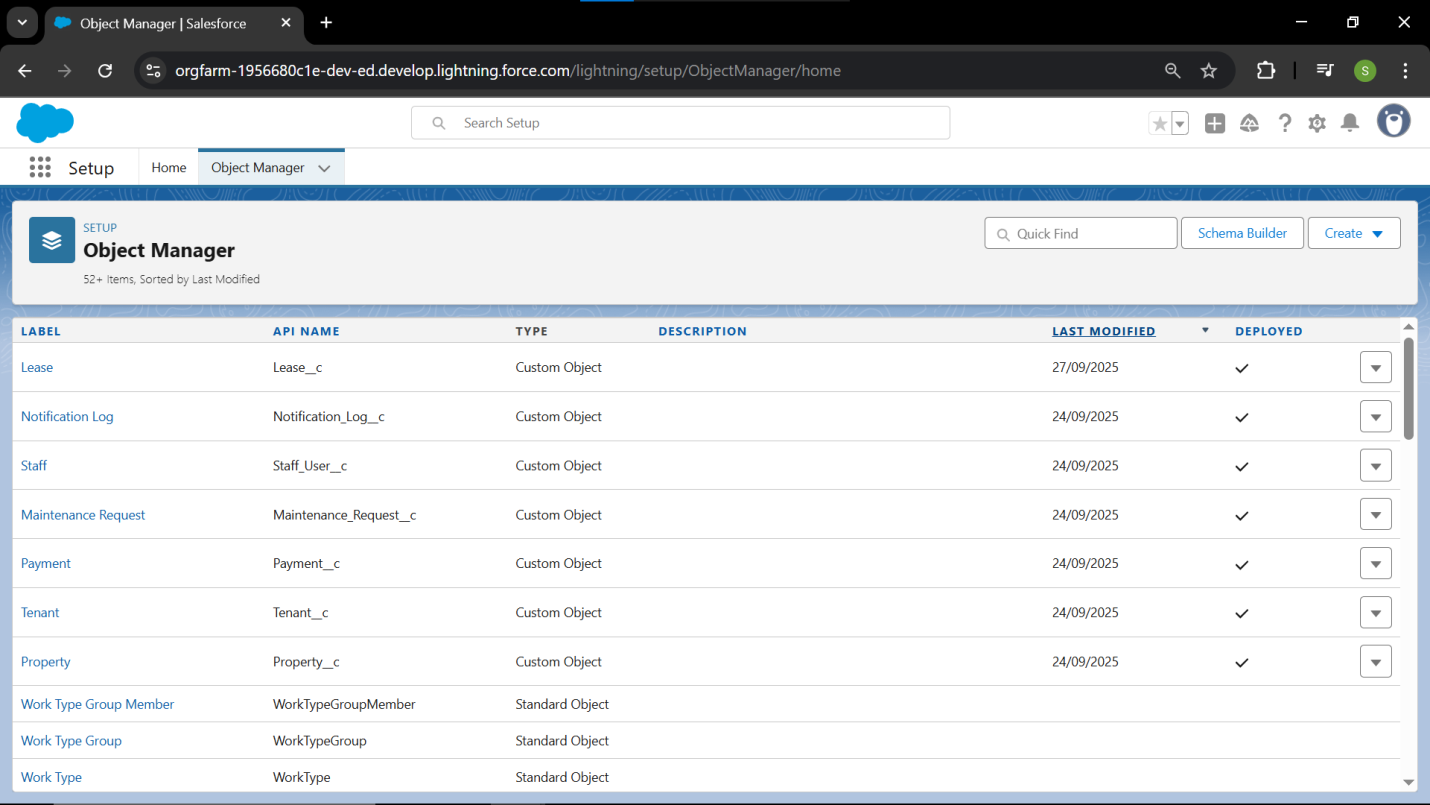
* **Action:** Log into your clean Developer Edition org.
* **Step:** Click the **gear icon** ⚙️ → **Setup** → Quick Find: Object Manager.
* **Action:** Click **Create** → **Custom Object**.

*Repeat the following steps for all seven custom objects:*

| Custom Object | Label | Record Name Type | Key OWD Setting |
| --- | --- | --- | --- |
| **1.** | Property | Text | Private |
| **2.** | Tenant | Text | Private |
| **3.** | Lease | Text | Private |
| **4.** | Payment | Text | Controlled by Parent |
| **5.** | Maintenance Request | Text | Private |
| **6.** | Staff/User | Text | Private |
| **7.** | Notification Log | Text | Private |

Export to Sheets

* **Step:** For each object, set **Record Name** to [Object Name] Name (as **Text**).
* **Step:** Check the boxes for **Allow Reports** and **Allow Search**.
* **Step:** Click **Save** after each one.

****

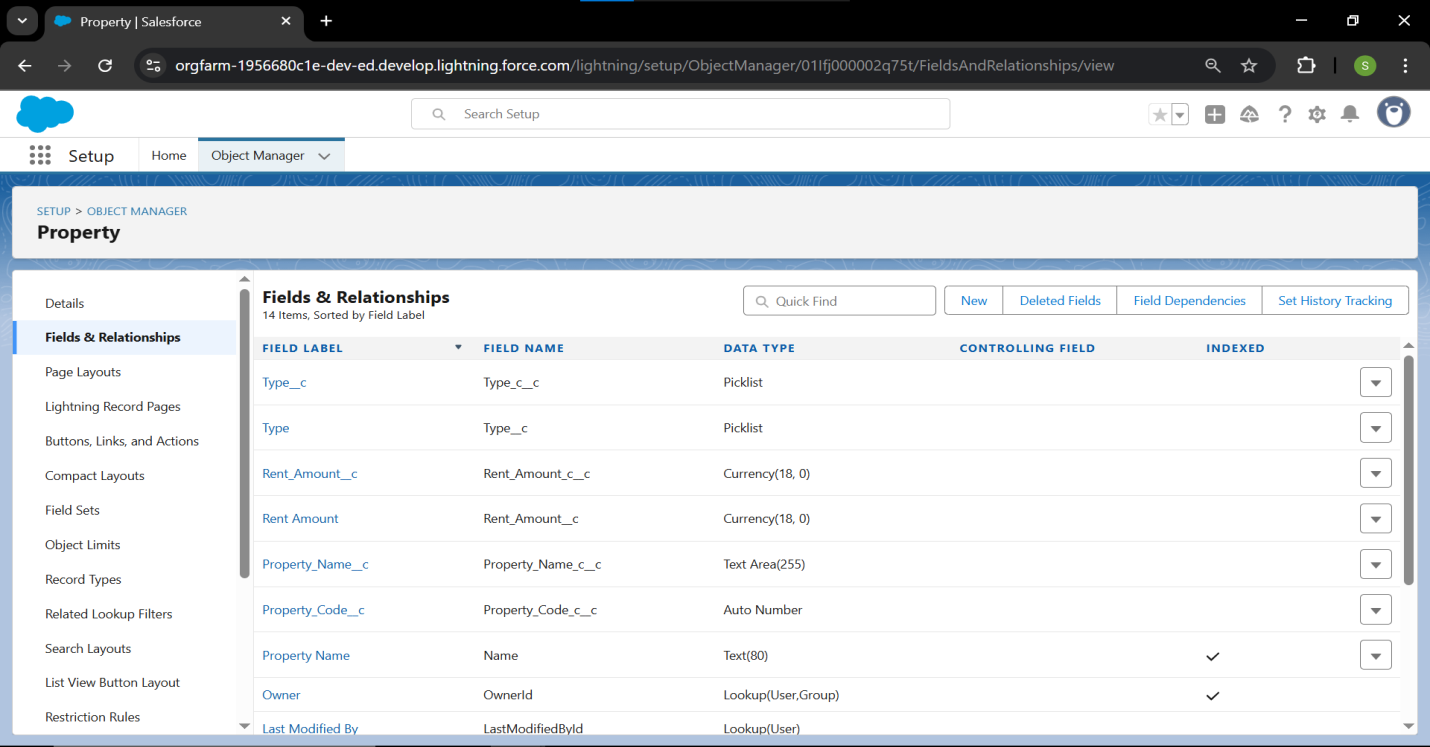
### Fields: Documentation Details 📝

* **Primary Objective:** To define the data attributes needed by the business, ensuring proper data types are selected for financial, date, and text fields.
* **Key Field Types Implemented:**
  1. **Financial:** Currency fields were used for all rent, payment, and late fee amounts.
  2. **Date/Time:** Specific Date fields track lifecycle milestones (Lease Start/End Date, Due Date, Resolution Date).
  3. **Controlled Input:** Numerous Picklist fields were created (Lease Status, Issue Type, Payment Status) to ensure data consistency for reporting and automation.
* **Security Impact:**
  1. **Field-Level Security (FLS)** was configured during creation to ensure that restricted users (like Tenants and Maintenance Staff) only see the fields relevant to their role.

### Fields: Implementation Steps 🛠️

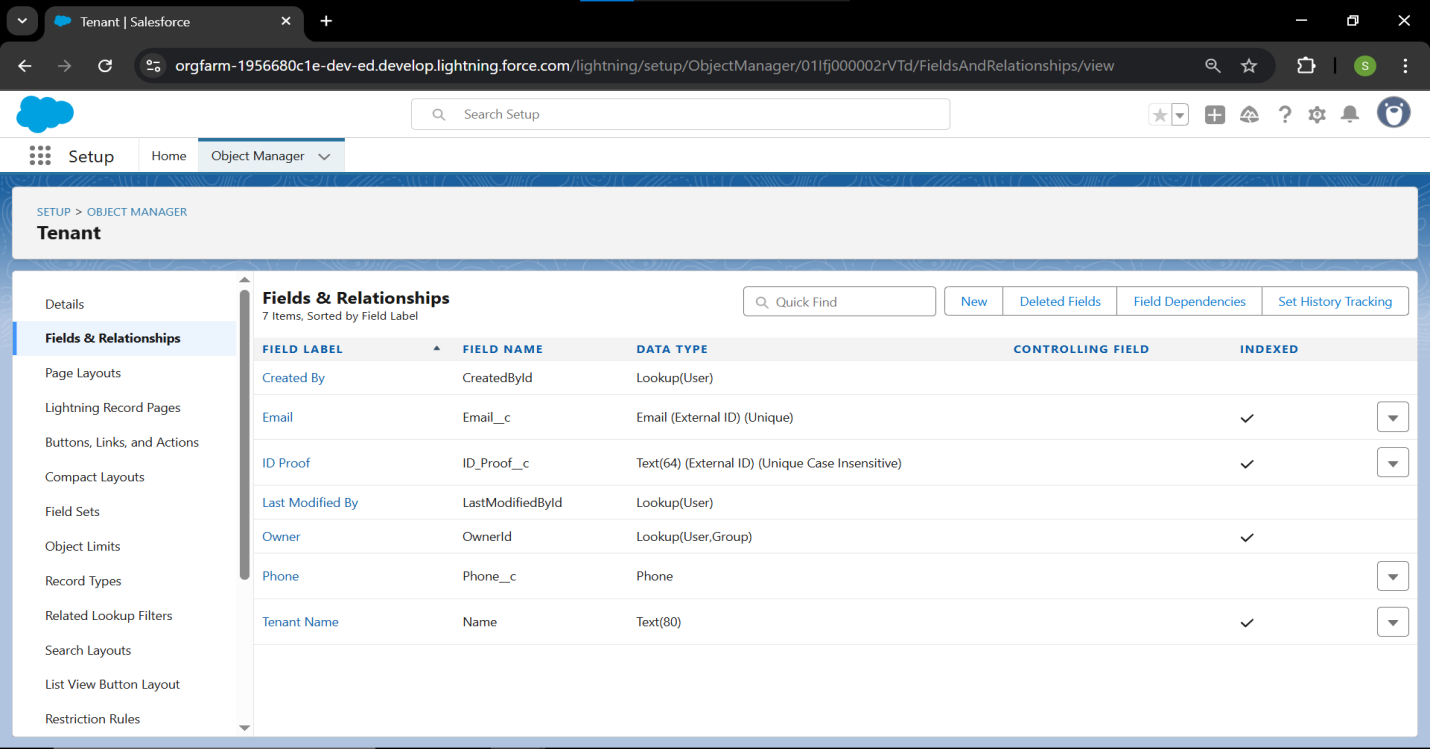
#### 1. Property Object Fields

* **Step 1.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Property** custom object.
  + Select **Fields & Relationships** and click the **New** button.
* **Step 1.2: Create Address Field**
  + **Action:** Select Data Type: **Text Area (Long)**. Click **Next**.
  + **Action:** Field Label: Address. Click **Next**.
  + **Action:** Set FLS (Field-Level Security): Ensure **Visible** is checked for **RentEase Property Manager** and **Read Only** for **RentEase Maintenance Staff**.
  + **Action:** Click **Next** and check the box to add the field to the page layout. Click **Save & New**.
* **Step 1.3: Create Type Field**
  + **Action:** Select Data Type: **Picklist**. Click **Next**.
  + **Action:** Field Label: Type.
  + **Action:** Enter values, using a separate line for each: Residential, Commercial.
  + **Action:** Click **Next**, set FLS, and click **Save & New**.
* **Step 1.4: Create Rent Amount Field**
  + **Action:** Select Data Type: **Currency**. Click **Next**.
  + **Action:** Field Label: Rent Amount.
  + **Action:** Click **Next**, set FLS, and click **Save & New**.
* **Step 1.5: Create Availability Status Field**
  + **Action:** Select Data Type: **Picklist**. Click **Next**.
  + **Action:** Field Label: Availability Status.
  + **Action:** Enter values: Available, Occupied, Under Maintenance.
  + **Action:** Click **Next**, set FLS, and click **Save**. *(We click Save here, not Save & New)*



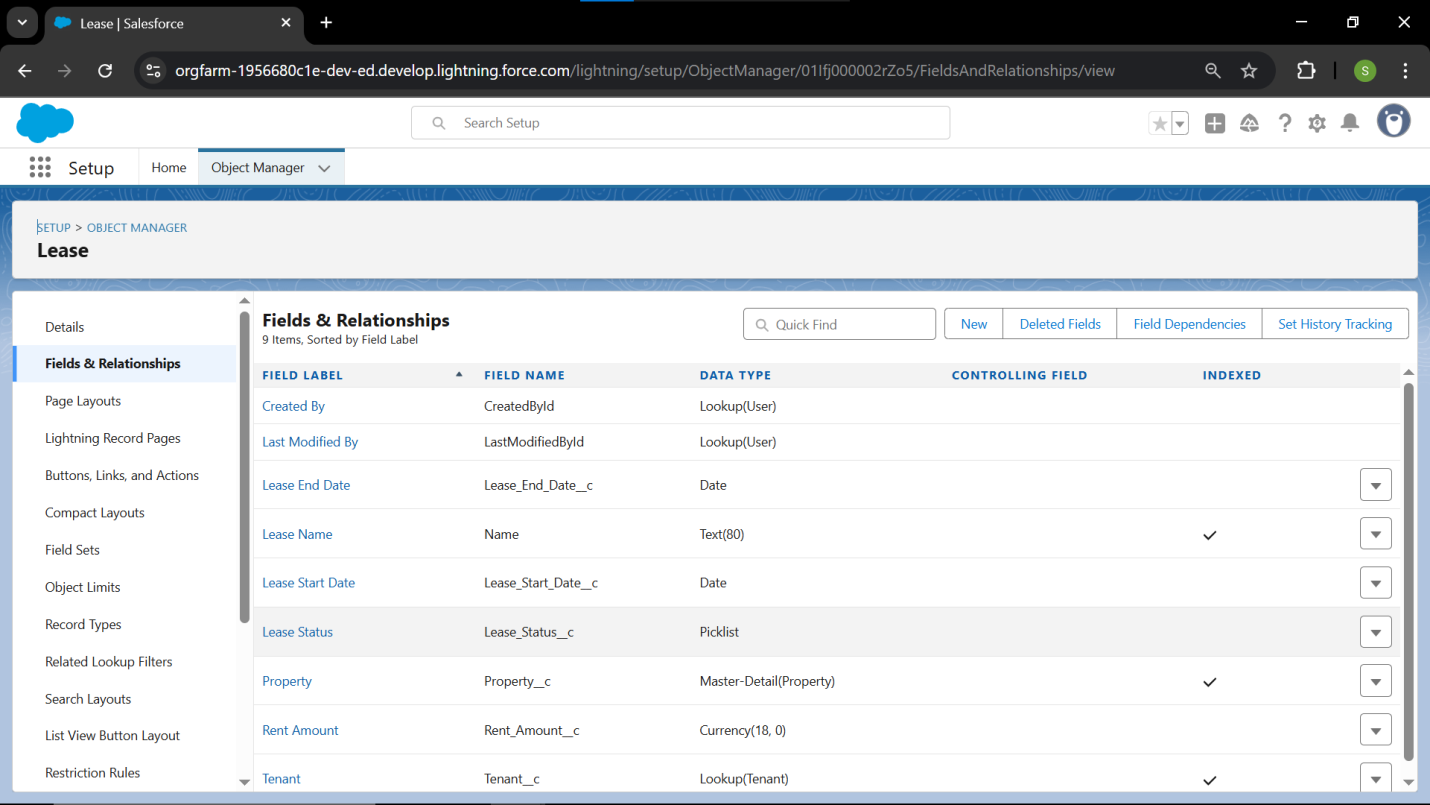
#### 2. Tenant Object Fields

* **Step 2.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Tenant** custom object.
  + Select **Fields & Relationships** and click **New**.
* **Step 2.2: Create Fields** (Use **Save & New** after the first two, and **Save** after the last)
  + **Field 1:** **Email** (Data Type: **Email**). Label: Email.
  + **Field 2:** **Phone** (Data Type: **Phone**). Label: Phone.
  + **Field 3:** **ID Proof** (Data Type: **Text**). Label: ID Proof.



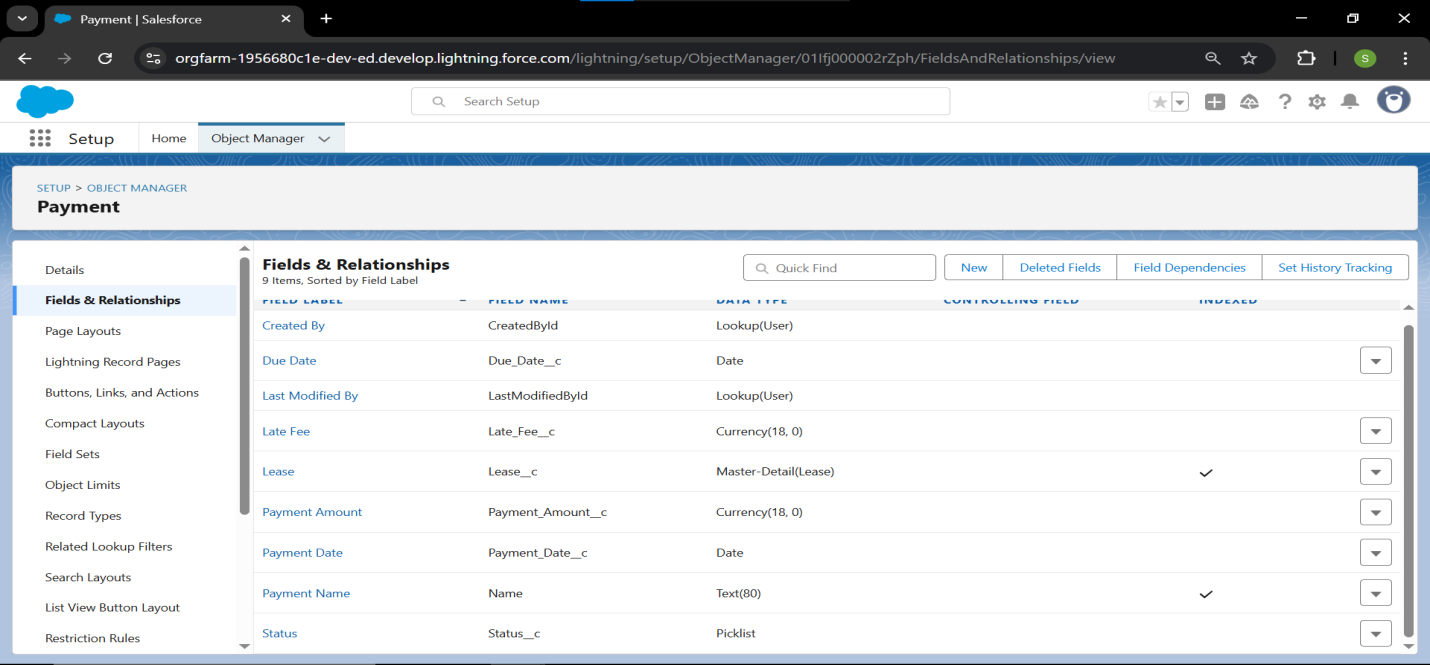
#### 3. Lease Object Fields

* **Step 3.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Lease** custom object.
  + Select **Fields & Relationships** and click **New**.
* **Step 3.2: Create Fields** (Use **Save & New** after the first three, and **Save** after the last)
  + **Field 1:** **Lease Start Date** (Data Type: **Date**). Label: Lease Start Date.
  + **Field 2:** **Lease End Date** (Data Type: **Date**). Label: Lease End Date.
  + **Field 3:** **Rent Amount** (Data Type: **Currency**). Label: Monthly Rent Amount.
  + **Field 4:** **Lease Status** (Data Type: **Picklist**). Label: Lease Status. Values: Draft, Active, Expired, Canceled.



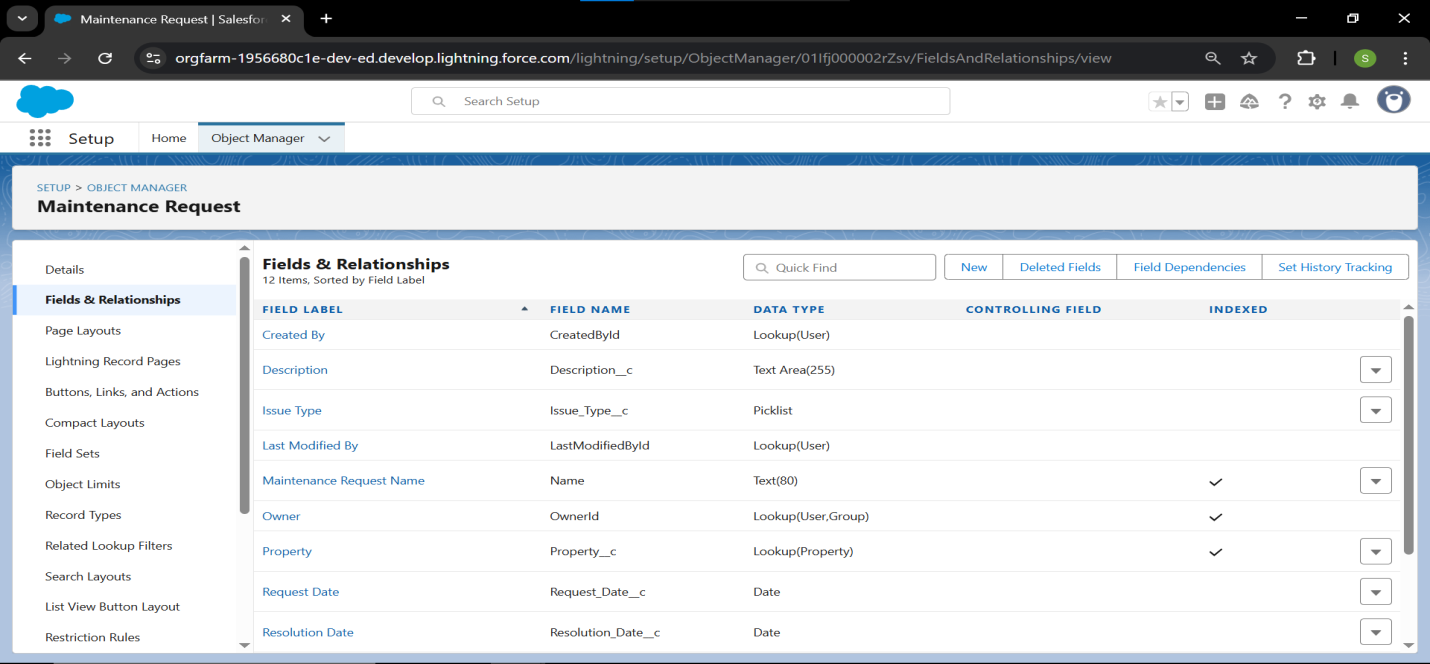
#### 4. Payment Object Fields

* **Step 4.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Payment** custom object.
  + Select **Fields & Relationships** and click **New**.
* **Step 4.2: Create Fields** (Use **Save & New** after the first four, and **Save** after the last)
  + **Field 1:** **Payment Amount** (Data Type: **Currency**). Label: Payment Amount.
  + **Field 2:** **Due Date** (Data Type: **Date**). Label: Due Date.
  + **Field 3:** **Payment Date** (Data Type: **Date**). Label: Payment Date.
  + **Field 4:** **Status** (Data Type: **Picklist**). Label: Status. Values: Pending, Paid, Overdue, Late Fee Applied.
  + **Field 5:** **Late Fee** (Data Type: **Currency**). Label: Late Fee Amount.



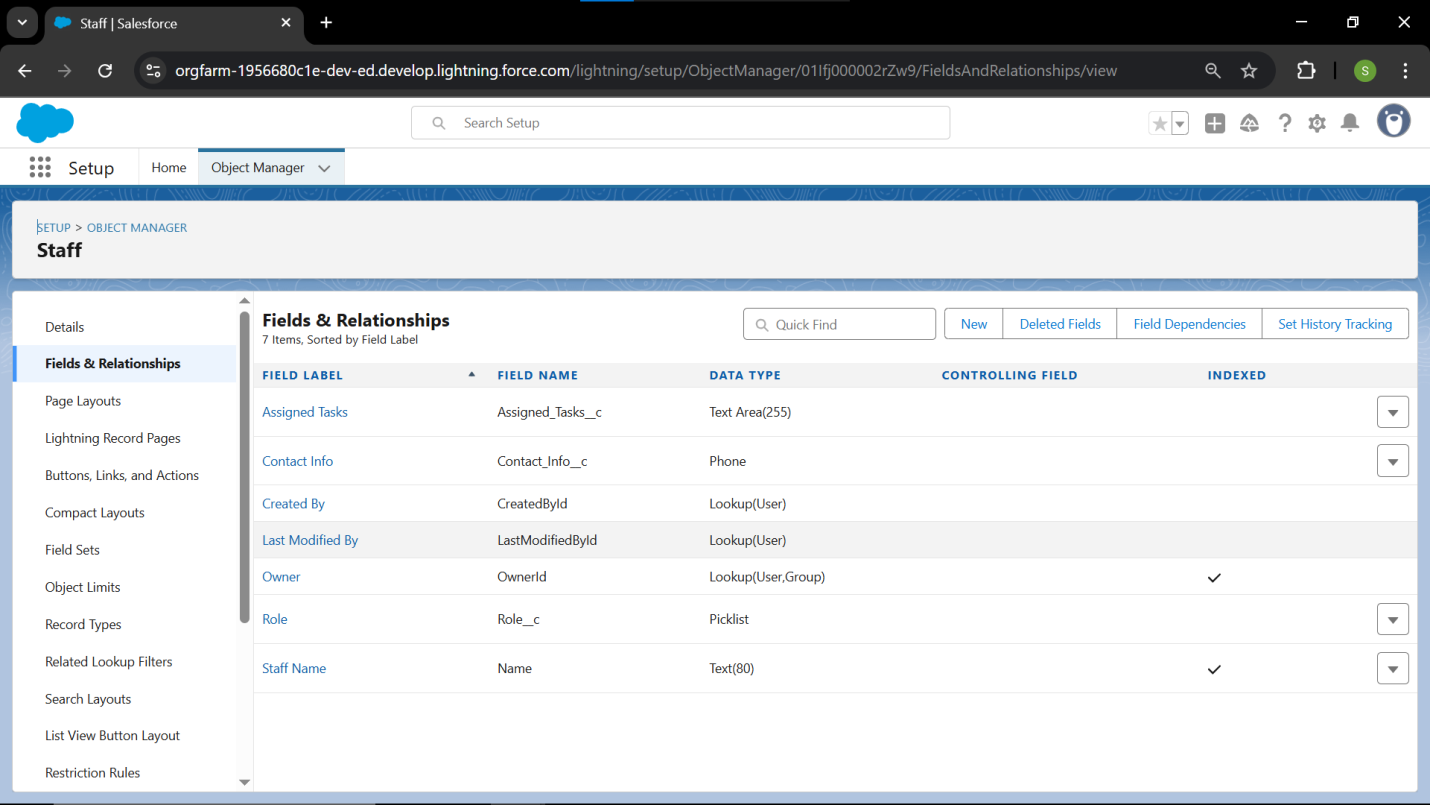
#### 5. Maintenance Request Fields

* **Step 5.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Maintenance Request** custom object.
  + Select **Fields & Relationships** and click **New**.
* **Step 5.2: Create Fields** (Use **Save & New** after the first four, and **Save** after the last)
  + **Field 1:** **Issue Type** (Data Type: **Picklist**). Label: Issue Type. Values: Plumbing, Electrical, Structural, Appliance.
  + **Field 2:** **Description** (Data Type: **Text Area (Long)**). Label: Description.
  + **Field 3:** **Status** (Data Type: **Picklist**). Label: Status. Values: Submitted, In Progress, On Hold, Completed.
  + **Field 4:** **Request Date** (Data Type: **Date**). Label: Request Date.
  + **Field 5:** **Resolution Date** (Data Type: **Date**). Label: Resolution Date.



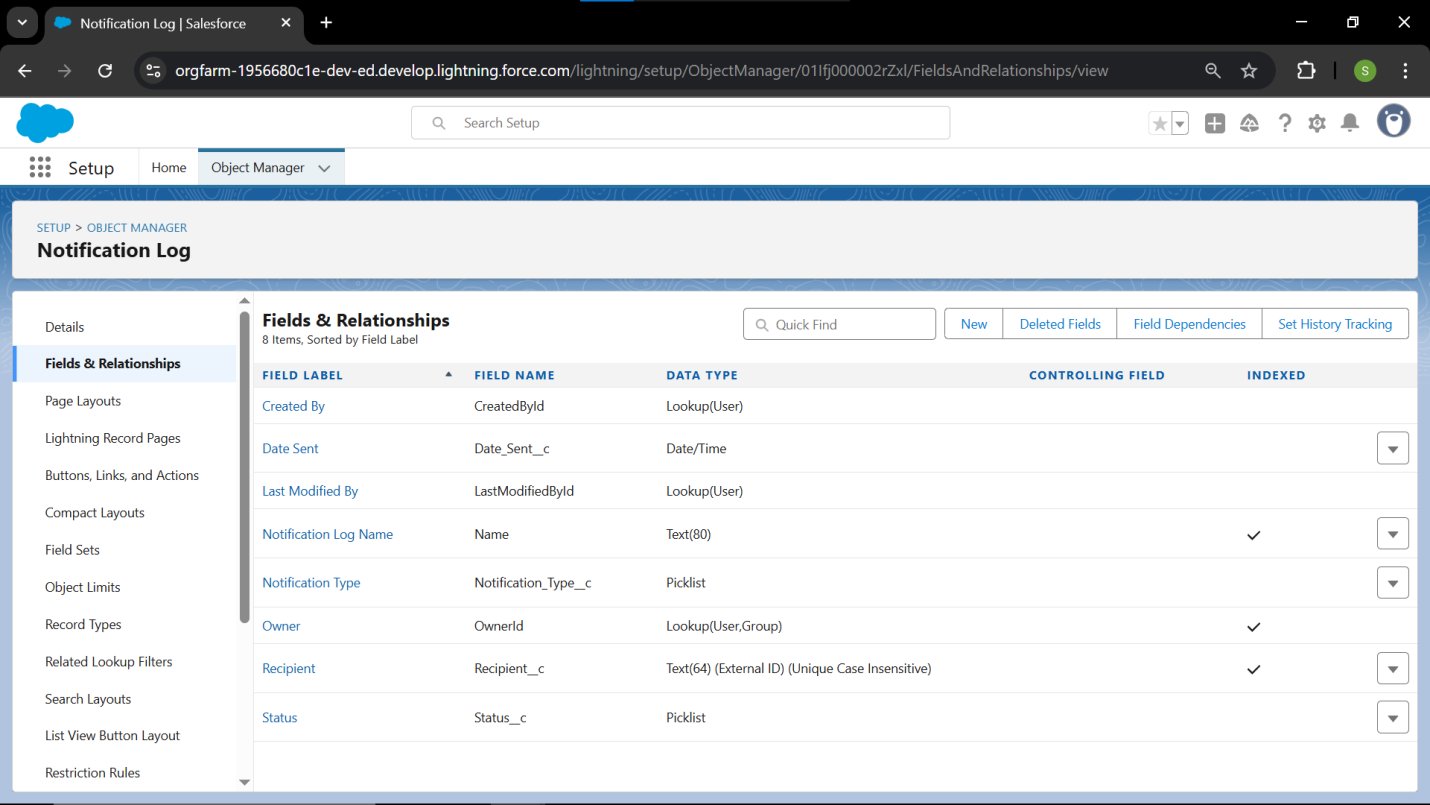
#### 6. Staff/User Object Fields

* **Step 6.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Staff/User** custom object.
  + Select **Fields & Relationships** and click **New**.
* **Step 6.2: Create Fields** (Use **Save & New** after the first two, and **Save** after the last)
  + **Field 1:** **Role** (Data Type: **Picklist**). Label: Role. Values: Property Manager, Maintenance Staff, Finance.
  + **Field 2:** **Contact Info** (Data Type: **Phone**). Label: Contact Info.
  + **Field 3:** **Assigned Tasks** (Data Type: **Text Area**). Label: Assigned Tasks Summary.



#### 7. Notification Log Fields

* **Step 7.1: Access Field Creation**
  + Go to **Object Manager** and click on the **Notification Log** custom object.
  + Select **Fields & Relationships** and click **New**.
* **Step 7.2: Create Fields** (Use **Save & New** after the first three, and **Save** after the last)
  + **Field 1:** **Notification Type** (Data Type: **Picklist**). Label: Notification Type. Values: Rent Due, Lease Renewal, Maintenance Update.
  + **Field 2:** **Recipient** (Data Type: **Text**). Label: Recipient Identifier.
  + **Field 3:** **Date Sent** (Data Type: **Date/Time**). Label: Date Sent.
  + **Field 4:** **Status** (Data Type: **Picklist**). Label: Delivery Status. Values: Sent, Delivered, Failed.



### Record Types Details

* **Primary Objective:** To enable process bifurcation and user interface specialization on the **Property** object, ensuring data integrity and relevance based on the property type.
* **Types Created and Purpose:**
  1. **Residential Property:** Created to manage all residential assets.
  2. **Commercial Property:** Created to manage all commercial assets.
  3. This step lays the groundwork for different validation rules or automation unique to each type.
* **Picklist Control:**
  1. The custom Type picklist field is now **controlled by the Record Type**.
  2. Users creating a record are restricted to selecting only the relevant type (Residential ↔ 'Residential', Commercial ↔ 'Commercial').

### Record Types: Steps 🛠️

#### 1. Access Record Type Creation (Property Object)

* **Step:** Go to **Object Manager** and click on the **Property** custom object.
* **Step:** In the sidebar, select **Record Types**.
* **Step:** Click the **New** button.

#### 2. Create 'Residential Property' Record Type

* **Action:** Select **Master** from the existing **Record Type** dropdown to copy from (or the current default).
* **Action:** Set the **Record Type Label** to Residential Property.
* **Action:** Set the **Record Type Name** to Residential\_Property.
* **Action:** Ensure the Record Type is **Active** (check the box).
* **Action:** Enable the Record Type for all four custom profiles you created by checking the 'Enabled' box for each.
* **Action:** Set the **Default Record Type** for the **RentEase Property Manager** Profile to **Residential Property**.
* **Step:** Click **Next**.

#### 3. Assign Picklist Values (Filtering)

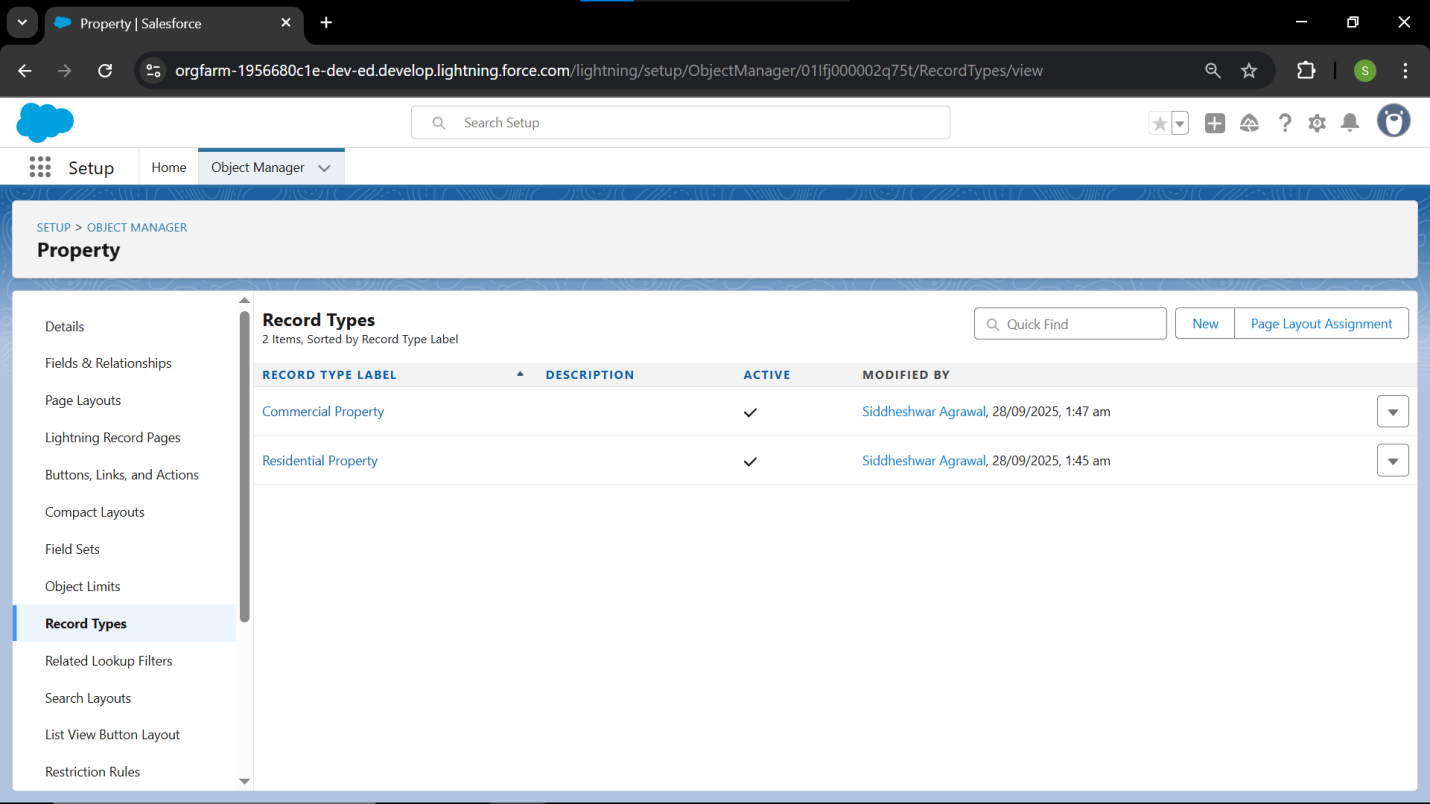
* **Action:** You will be redirected to the page for **Property**'s picklist value assignment.
* **Action:** Locate the **Type** field (which has values 'Residential' and 'Commercial'). Click **Edit** next to it.
* **Action:** Under the 'Residential Property' list, move **'Commercial'** from **Selected Values** to **Available Values**.
* **Detail:** This ensures users creating a Residential record **only** see the 'Residential' value.
* **Step:** Click **Save**.

#### 4. Create 'Commercial Property' Record Type

* **Step:** Return to the **Record Types** page for the Property object and click **New**.
* **Action:** Select the newly created Residential Property record type to copy from.
* **Action:** Set the **Record Type Label** to Commercial Property.
* **Action:** Ensure the Record Type is **Active** and enabled for all four custom profiles.
* **Step:** Click **Next**.

#### 5. Assign Picklist Values (Filtering - Commercial)

* **Action:** You will again be taken to the picklist assignment page.
* **Action:** For the **Type** field, click **Edit**.
* **Action:** For this **Commercial Property** Record Type, move **'Residential'** from **Selected Values** to **Available Values**.
* **Detail:** This ensures users creating a Commercial record **only** see the 'Commercial' value.
* **Step:** Click **Save**.

****

### Page Layouts: Documentation Details 📝

* **Primary Objective:** To configure the user interface for optimal data entry and viewing efficiency, enforcing visual separation based on the data model.
* **Layouts Implemented:**
  1. **Dynamic Layouts:** Two dynamic layouts (**Residential Property Layout** and **Commercial Property Layout**) were created for the **Property** object.
  2. **Base Layouts:** Single custom layouts were created for all other six custom objects.
* **Layout Assignment Strategy:**
  1. Assignments for the **Property** object are **dynamic**, relying on the user's selection of a **Record Type** to determine the layout that loads.
  2. Custom fields were logically grouped into sections (e.g., Financial Details, Service Information) on each layout.

### Page Layouts: Implementation Steps 🛠️

#### 1. Create and Clone Page Layouts (Property Object)

* **Step:** Go to **Object Manager** and click on the **Property** custom object.
* **Step:** In the sidebar, select **Page Layouts**.
* **Action:** Click **Clone** next to the existing default **Property Layout**.
* **Action:** Name the new layout **Residential Property Layout**. Click **Save**.
* **Action:** Click **Clone** next to the original **Property Layout** again.
* **Action:** Name the second new layout **Commercial Property Layout**. Click **Save**.

#### 2. Customize Layouts (Residential Example)

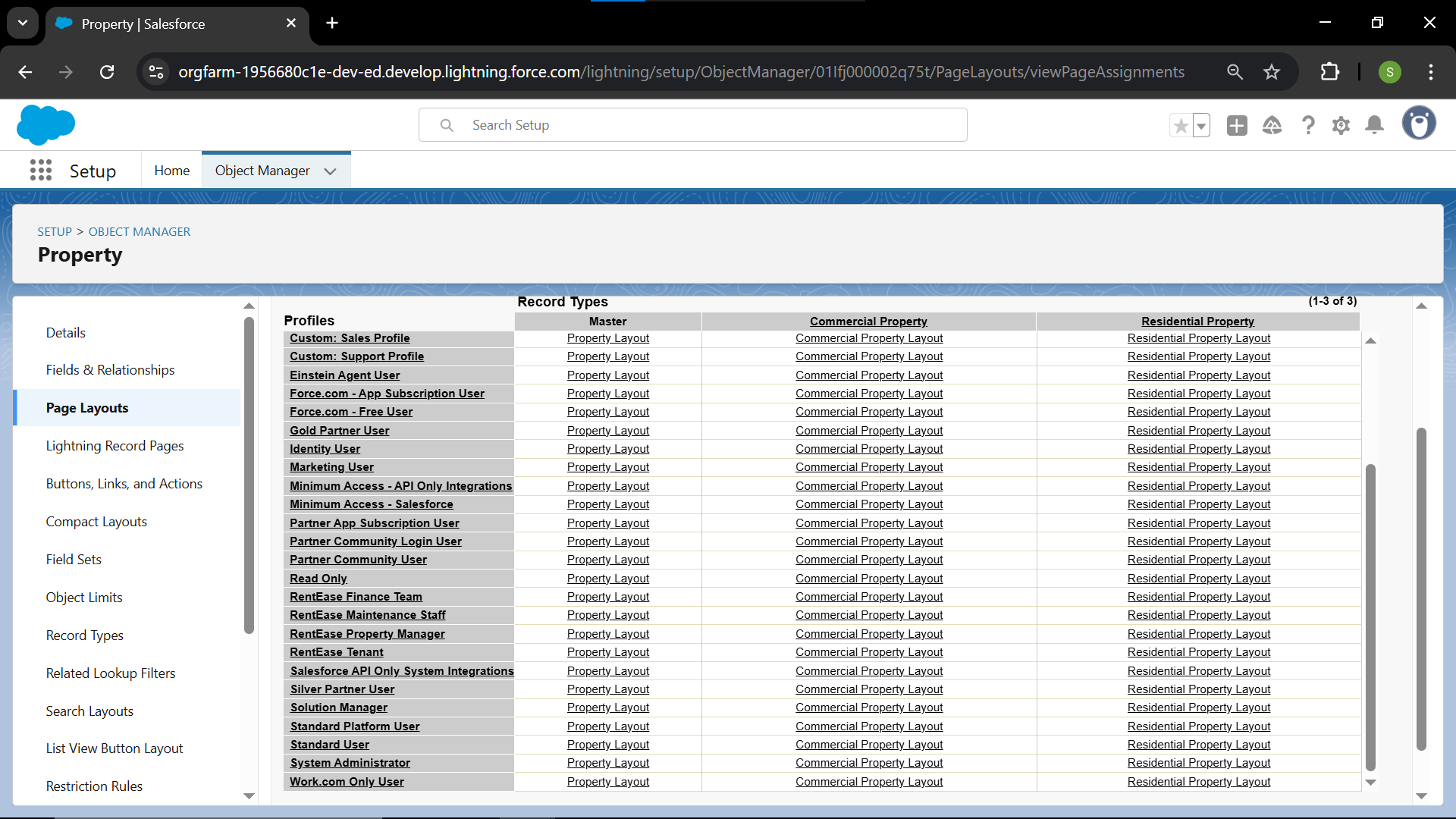
* **Step:** Click on the **Residential Property Layout** name to edit it.
* **Action:** **Organize Fields:** Drag the Address field into the main information section. Group similar custom fields into new sections (e.g., drag Rent Amount into a new section named Financial Details).
* **Action:** **Organize Related Lists:** Scroll down to the Related Lists section. Drag the **Leases** and **Maintenance Requests** lists to the top of the section, as they are the most critical data.
* **Step:** Click **Save**.
* **Step:** Repeat the organization process for the **Commercial Property Layout**, ensuring any commercial-specific fields (if added later) are prominently featured.

#### 3. Assign Layouts Dynamically (Property Object)

* **Step:** Return to the **Page Layouts** section for the Property object.
* **Step:** Click the **Page Layout Assignment** button.
* **Action:** Click **Edit Assignment**.
* **Action:** For the **RentEase Property Manager** Profile:
  + Select the **Residential Property Layout** for the **Residential Property** Record Type column.
  + Select the **Commercial Property Layout** for the **Commercial Property** Record Type column.
* **Action:** Repeat this assignment for the other internal profiles (**Finance Team, Maintenance Staff**), ensuring the assignments match their permissions.
* **Step:** Click **Save**.

#### 4. Create Base Layouts for Remaining Objects

* **Action:** Repeat the process of navigating to the following objects and creating one basic, clean layout for each (as they do not currently have Record Types): **Tenant, Lease, Payment, Maintenance Request, Staff/User,** and **Notification Log**.
* **Action:** For each layout, drag custom fields to the main content area and organize related lists logically.
* **Action:** Assign this single new layout to all custom profiles for that specific object.

****

### Compact Layouts: Documentation Details 📝

* **Primary Objective:** To optimize the record highlights panel and mobile user experience by displaying the most crucial information at a glance.
* **Layout Definition:**
  1. A dedicated **Compact Layout** was created for all seven custom objects.
  2. Each layout contains the most business-critical fields (e.g., Rent Amount, Status, End Date) necessary for quick recognition without navigating into the full record detail.
* **Assignment Strategy:**
  1. The newly created layouts were assigned as the **Primary Compact Layout** for their respective objects.
  2. This ensures the condensed view is automatically applied to the record highlights panel in Lightning Experience and the Salesforce Mobile App.

### Compact Layouts: Implementation Steps 🛠️

#### 1. Create Compact Layout (Property Object)

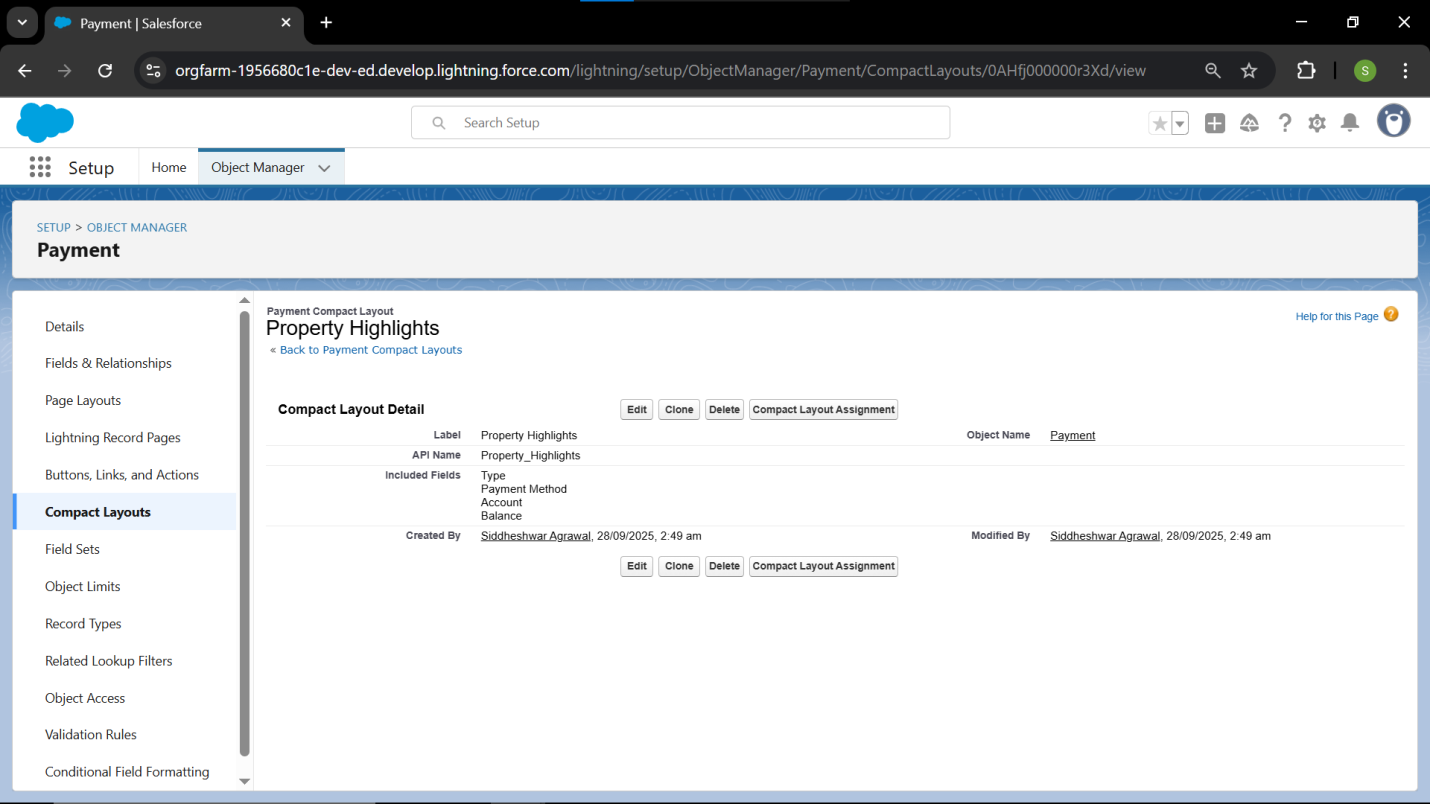
* **Step:** Go to **Object Manager** and click on the **Property** custom object.
* **Step:** In the sidebar, select **Compact Layouts**.
* **Step:** Click the **New** button.
* **Action:** Set the **Label** to Property Highlights. The **Name** will auto-fill.
* **Action:** In the **Available Fields** section, add the following key fields, in this exact order, to the **Selected Fields** list:
  1. Property Name (Auto-added or use the custom name field)
  2. Type
  3. Rent Amount
  4. Availability Status
  5. Address
* **Detail:** We prioritize these five fields because the top four to six fields are the most visible in the Lightning Highlights Panel and mobile views.
* **Step:** Click **Save**.

#### 2. Assign Compact Layout (Property Object)

* **Step:** While still on the **Compact Layouts** page, click the **Compact Layout Assignment** button.
* **Step:** Click **Edit Assignment**.
* **Action:** In the **Primary Compact Layout** dropdown, select the newly created **Property Highlights** layout.
* **Detail:** This sets the new layout as the default for all users and all Record Types for the Property object.
* **Step:** Click **Save**.

#### 3. Create Compact Layouts for Remaining Core Objects

* **Action:** Repeat the process for the remaining core custom objects, creating one new layout for each:
  + **Lease Object:** Create layout Lease Highlights. Include: Lease Name, Property (Lookup), Tenant (Lookup), Lease Status, Lease End Date.
  + **Payment Object:** Create layout Payment Highlights. Include: Payment Name, Lease (Master-Detail), Due Date, Payment Amount, Status.
  + **Maintenance Request Object:** Create layout Request Highlights. Include: Request Name, Property (Lookup), Tenant (Lookup), Issue Type, Service Status.
* **Action:** For each of these layouts, ensure you assign it as the **Primary Compact Layout**.

****

**Schema Builder: Details 📝**

* **Primary Objective:** To visually inspect and confirm the health and integrity of the data model schema after fields and relationships have been created.
* **Verification Checkpoints:**
  1. **Visual Confirmation:** The **Schema Builder** was used as a diagnostic tool to map the current state of the data model.
  2. **Relationship Integrity:** The two critical **Master-Detail Relationships** (Property → Lease and Lease → Payment) were visually confirmed to ensure proper security inheritance and OWD compliance.
  3. **Field Presence:** Verified that all custom fields are correctly attached to their respective objects.

### Schema Builder: Steps 🛠️

#### 1. Access the Schema Builder Tool

* **Step:** Click the **gear icon** ⚙️ and navigate to **Setup**.
* **Step:** Use Quick Find to search for and select **Schema Builder**.
* **Detail:** The tool will open in a new tab, initially showing a blank canvas or all available objects.

#### 2. Filter and Display Relevant Objects

* **Step:** In the **Elements** sidebar, click the **Clear All** button at the top of the object list to empty the canvas.
* **Action:** Click the **Object Settings** filter (or the dropdown filter).
* **Action:** Select the following objects to display them on the canvas:
  + **All 7 Custom Objects** (Property, Lease, Payment, etc.)
  + **Account** (Standard, for Owner tracking)
  + **Contact** (Standard, for Tenant linkage)
* **Step:** Click **View Options** and deselect **Show Hidden Elements** to ensure a clean visual.
* **Step:** Click **Auto-Layout** to organize the objects neatly.

#### 3. Verify Relationships (Critical Check)

* **Action:** Visually inspect the lines connecting the objects:
  + Verify the line between **Lease** and **Payment** is **thick** (Master-Detail).
  + Verify the line between **Property** and **Lease** is **thick** (Master-Detail).
  + Verify the lines for **Tenant**, **Maintenance Request**, and **Staff/User** are **thin** (Lookup).

#### 4. Verify Field Existence (Spot Check)

* **Step:** Click the **gear icon** on the **Property** object card.
* **Action:** Click **Show Fields** and confirm all your custom fields (Rent Amount, Availability Status, Type) are present.
* **Step:** Repeat the field check for the **Payment** and **Maintenance Request** objects.

**Relationship Types: Documentation Details 📝**

* **Primary Objective:** To formally define the purpose and functional impact of each chosen relationship type on data security, ownership, and integrity within the schema.

**1. Master-Detail Relationships (Strongest Link)**

* **Definition:** Chosen where data ownership, security, and lifecycle management are strictly dependent on the parent record.
* **Use Cases in RentEase:**
  1. **Property → Lease:** A Lease is inextricably linked to one Property. If the Property were deleted (in a theoretical scenario), all related Lease records would be deleted automatically (cascade delete).
  2. **Lease → Payment:** A Payment is fiscally meaningless without a Lease contract. This link enforces data integrity and makes the **Payment** object's security follow the parent **Lease** (Controlled by Parent OWD).
* **Security Impact:** Deleting the Master record automatically deletes all Detail records. The Detail record inherits the Master's sharing and security settings.

**2. Lookup Relationships (Flexible Link)**

* **Definition:** Chosen where two objects need to be linked for reference, but deletion of one record should **not** affect the other, or where the relationship is optional.
* **Use Cases in RentEase:**
  1. **Lease → Tenant:** A Tenant record (linked to a Contact) can exist independently of any Lease. The relationship is flexible and non-restrictive.
  2. **Maintenance Request → Staff/User:** A Request must reference the assigned Staff member, but deleting the Request does not affect the Staff/User record (and vice versa).
* **Security Impact:** Lookup fields do not affect security; access to the child record is independent of the parent.

**3. Hierarchical Relationship**

* **Definition:** A specialized lookup relationship that is **only available on the User object** and allows users to link to other users (e.g., Manager field).
* **Use Case in RentEase:**
  1. **Not Used for Custom Objects:** This relationship type is **not applicable** to any of the seven custom objects (Property, Lease, etc.).
  2. The management structure is already handled by the built-in **Role Hierarchy**, which performs the same function (defining a reporting structure) for sharing.

### Junction Objects: Documentation Details 📝

* **Primary Objective:** To correctly model a Many-to-Many relationship by introducing a third object that links two Master objects.
* **M:N Relationship Model:**
  1. **Masters:** The **Tenant** object and the **Amenity** custom object.
  2. **Junction Object:** The **Tenant Amenity Link** object.
  3. This structure allows a single **Tenant** record to be linked to multiple **Amenity** records, and vice versa.
* **Security Impact:**
  1. The Junction Object has **two Master-Detail relationships**, meaning its security and data deletion rules are inherited from **both** the **Tenant** and **Amenity** master records.
  2. The resulting sharing access is calculated by the most restrictive setting between the two masters.

### Junction Objects: Implementation Steps 🛠️

#### 1. Identify the M:N Relationship

* **Need:** A single **Tenant** may use multiple **Amenities** (e.g., parking, storage unit), and a single **Amenity** (e.g., Parking Lot B) is linked to multiple **Tenants**.
* **Action:** We need a third object (the Junction Object) to sit between the two Masters: **Tenant** and a new **Amenity** object.

#### 2. Create the Second Master Object (Amenity)

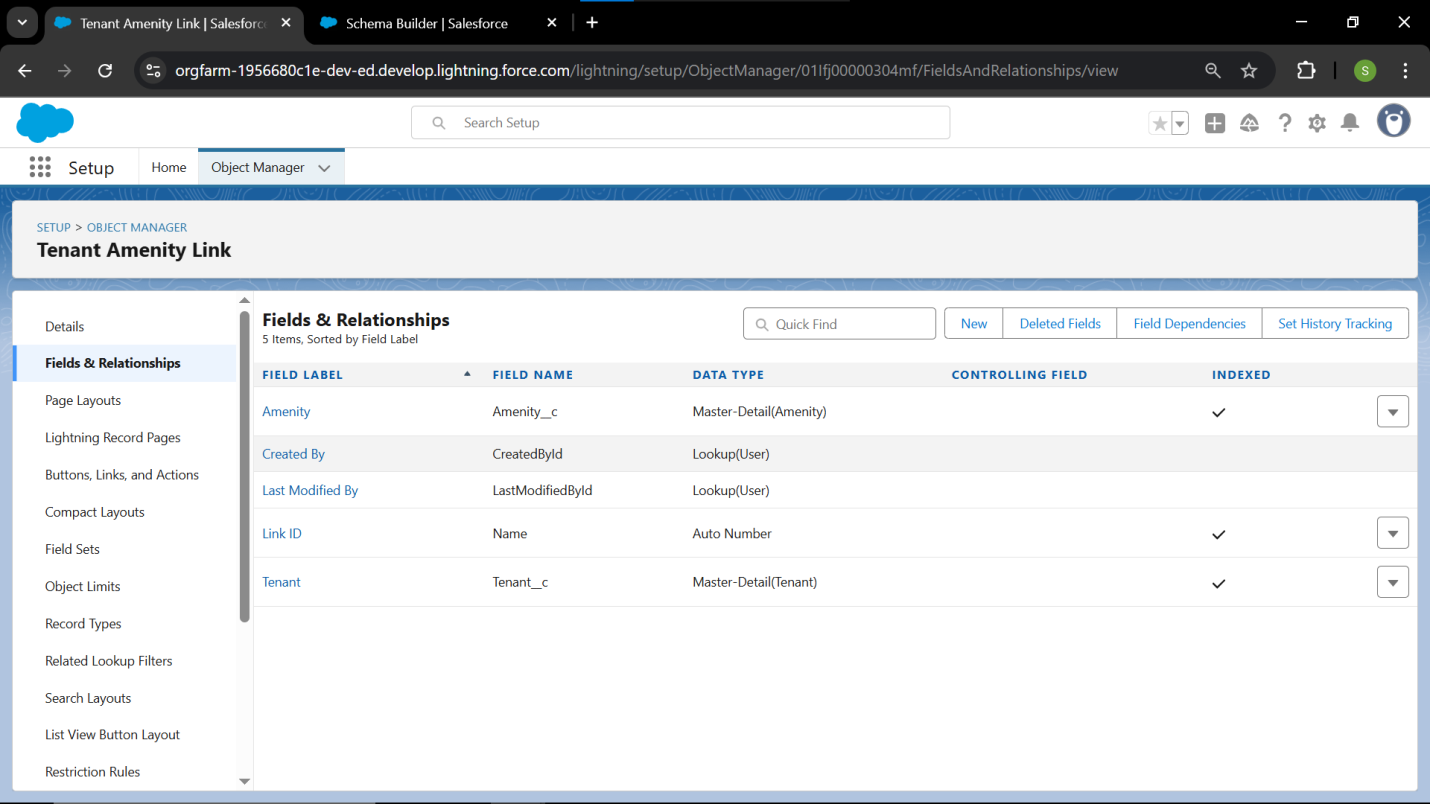
* **Step:** Go to **Object Manager** and click **Create** → **Custom Object**.
* **Action:** Set **Label** to Amenity, **Plural Label** to Amenities.
* **Action:** Set **Record Name** to Amenity Name (Text).
* **Step:** Click **Save**.

#### 3. Create the Junction Object (Tenant Amenity Link)

* **Step:** Click **Create** → **Custom Object**.
* **Action:** Set **Label** to Tenant Amenity Link, **Plural Label** to Tenant Amenity Links.
* **Action:** Set **Record Name** to Link ID (Auto-Number).
  + **Detail:** Using Auto-Number for the Junction Object's name is a best practice, as the record is purely for linking data.
* **Step:** Click **Save**.

#### 4. Establish the Two Master-Detail Relationships

* **Step:** While on the **Tenant Amenity Link** object, go to **Fields & Relationships** and click **New**.
* **Action (Link 1 - Tenant):** Select **Master-Detail Relationship** → **Related To:** Tenant.
  + **Detail:** This makes the **Tenant** object the first master.
  + **Step:** Click **Next** through FLS and Layouts, then click **Save & New**.
* **Action (Link 2 - Amenity):** Select **Master-Detail Relationship** → **Related To:** Amenity.
  + **Detail:** This makes the **Amenity** object the second master.
  + **Step:** Click **Next** through FLS and Layouts, then click **Save**.

****

### External Objects: Documentation Details 📝

* **Primary Objective:** To define the strategy for integrating data from external, non-Salesforce systems (e.g., a legacy accounting system or external inventory) without requiring data migration.
* **Definition:** An External Object is a way to reference data that lives outside of your Salesforce database, making it appear and function like a standard custom object using **Salesforce Connect**.

#### 1. Hypothetical Use Case: Legacy Accounting System

* **Need:** The RentEase Finance Team needs to view a tenant's complete **5-Year Transaction History** from a legacy ERP system, but the data volume is too large to import into Salesforce.
* **Solution:** Create an **External Object** named Legacy\_Transaction\_\_x that pulls the data on demand from the external system via an API call.

#### 2. Strategy and Configuration Overview

* **Prerequisite:** The organization must have the necessary **Salesforce Connect License** (not available in Dev Org).
* **Step 1: External Data Source:** A named connection (using protocols like OData or custom APIs) must be established in Salesforce to point to the external system.
* **Step 2: External Object Creation:** The External Object (Legacy\_Transaction\_\_x) is created in the Salesforce UI.
* **Step 3: External Fields:** Fields are mapped to the columns in the external database table. The standard record ID in the external system is used as the **External ID** in the External Object.
* **Step 4: Relationship:** A Lookup relationship is created from the custom **Lease** object to the **External Object** to show the transaction history on the Lease record page.

#### 3. Benefits of External Objects

* **Data Integrity:** Data remains live in the source system, ensuring users always see the latest transaction history.
* **Storage Optimization:** Saves on Salesforce data storage limits, as the data is never physically moved into the org.
* **Real-Time Access:** Data is accessed in real-time when the user clicks on the related list.
* **Status:** The External Object strategy is defined, concluding **Phase 3: Data Modeling & Relationships**.