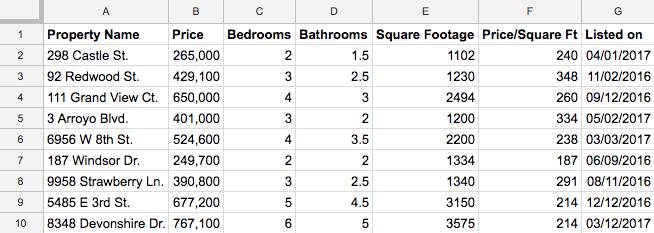
**Data Modelling**

**What is Data Modelling?**

Data Model is collection of conceptual tools for describing data, data relationships.

**Need for Data model**



In Salesforce, we think about database tables as **objects**, we think about columns as **fields**, and rows as **records**.

So instead of an account spreadsheet or table, we have an Account object with fields and a bunch of identically structured records.

When we talk about the data model, we’re talking about the collection of objects and fields in an app.

**Objects**

**Object reprents a Table, Which is used to store the Application / Business specific data.**

* Salesforce supports several different types of objects. There are standard objects, custom objects, external objects, platform events, and Big Objects
* **Standard objects** are objects that are included with Salesforce.
* **Salesforce provides a set of readymade objects as below.**

**Ex: Campaign, Lead, Account, Contact, Opportunity, Case, Solution, Order, Contract,Quote, PriceBook, Feedback,....etc.**

* Common business objects like Account, Contact, Lead, and Opportunity are all standard objects.

**Custom objects** are objects that you create to store information that’s specific to your company or industry.

Objects are containers for your information, but they also give you special functionality. For example, when you create a custom object, the platform automatically builds things like the page layout for the user interface.

**Creating a Custom Object:**

1. Click the gear icon The setup gear. at the top of the page and launch setup.
2. Click the **Object Manager** tab.
3. Click **Create** | **Custom Object** in the top-right corner.
4. For Label, enter [**CustomerPolicyDetails**](https://app-3c-dev-ed.lightning.force.com/one/one.app#/setup/ObjectManager/01I5g000003EM8v/view).
5. For Plural Label, enter [CustomerPolicyDetails](https://app-3c-dev-ed.lightning.force.com/one/one.app#/setup/ObjectManager/01I5g000003EM8v/view).
6. Prior to saving the custom object, scroll to the bottom of the page and select the checkbox **Launch New Custom Tab Wizard after saving this custom object**.
7. Leave the rest of the values as default and click **Save**.
8. On the New Custom Object Tab page, click the Tab Style field and select a style you like. The style sets the icon to display in the UI for the object.
9. Click **Next**, **Next**, and **Save**.

**Object ID / Schema Id / Object Key-Prefix:**

**Salesforce provides a set of readymade objects by default as part of Salesforce CRM Application.**

**Salesforce allocates an Unique Identification Number for each object called as "Object Id / Schema ID", Which is a 3-Characters Alphanumerical Number as below.**

**Object Name Object ID**

**--------------------------------------**

**Campaign ---> 701**

**Lead ---> 00Q**

**Account ---> 001**

**Contact ---> 003**

**Opportunity ---> 006**

**Case ---> 500**

**Solution ---> 501**

**Order ---> 801**

**Task ---> 00T**

**Event ---> 00U**

**Group ---> 00G**

**User ---> 005**

**Benefits:**

**1. By using Object Id, we can switch / jump from one object to another by placing the**

**object id inside the URL as below.**

**Syntax:**

**https://<DomainName>.Salesforce.com/<objectID>**

**Ex:**

**https://<DomainName.Salesforce.com>/500 ----> Jump to Case Object.**

**https://<DomainName.Salesforce.com>/005 ----> Jump to User Object.**

**https://<DomainName.Salesforce.com>/006 ----> Jump to Opportunity Object.**

**Note: Inside the URL, it will represent the first 15-Characters Record ID.**

**Last 3-Characters will not be visible inside the URL because of Security Reasons.**

**Note: We can get the Complete 18-Characters ID of the record by using SOQL Queries.**

**2. By using Object Id, we can recognize the Records inside the object.**

**Note: While creating the records inside the object, Salesforce allocates an**

**18-Characters Unique Identification Number for each record, called as "Record ID". Which is a Case-Sensitive ID.**

**Record ID : 18 - Characters Alphanumerical Number.**

**|----> First 3-Characters Represents ---> Object ID**

**| ----> Remaining 15-characters Represents the actual record id.**

**----> Last 3-Characters represents**

**"CheckSum / Transaction Code"**

**Governor Limits:**

1. **In Free Developer Edition**:

We can create max. of 400 Custom Objects.

2. **In Unlimited Edition**:

We can create max. of 2,000 Custom Objects.

**While Creating a Custom Object, Salesforce provides the below features by default.**

**Ex:**

**Custom Object : Customer.**

**1. Table (Model) : Customer\_\_C**

**(Provides 5 Standard Fields:**

**Id, Name, Owner, CreatedBy, and LastModifiedBy)**

**2. Tab (GUI): Customers**

**3. Business Logic (Class) : Customer\_\_C**

**Fields:**

Every field has a data type. A data type indicates what kind of information the field stores. Salesforce supports a bunch of different data types, but here are a few you’ll run into.

**DataTypes:**

**While Creating a Custom field, we have to select the DataType. Which describes, What type of Data / Value can be store inside the field.**

**Note: Based on the Selected DataType, it will represent the corresponding**

**"Visual Appearance" on the User Interface.**

**Salesforce provides be below DataTypes, which can be selected upon creating the field.**

**1. Text:**

**Text DataType allows us to store max. of 255 Characters of Content inside the field, Which includes Alphanumerical Values and Special Characters.**

**While create a Text field, it will represent the Textbox on the user interface by default.**

**Properties:**

**1. Field Label:**

**This property is used to specify the Label / Static text to be visible on the User Interface.**

**2. Field Name:**

**It represents the actual column name inside the object.**

**3. Description:**

**To provides the Comments / Description about the field.**

**4. Help Text:**

**This property is used to specify the Tooltip text message to be visible to the user, upon placing the mouse on the field.**

**5. Length :**

**To specify the Number of Characters to be allowed in the field.**

**6. Required Checkbox:**

**To make the field required / mandatory.**

**Note: We have the below 5 ways to make a field required.**

**1. By Selecting "Required Checkbox", while creating field.**

**2. By using "Page Layout Customizations".**

**3. By using "Validation Rules".**

**4. By using "Apex Triggers".**

**5. By using "Visualforce Programming".**

**7. Unique Checkbox:**

**To make the field values unique. i.e. It won't allow the Duplicate values inside the field.**

**8. External ID Checkbox:**

**It allows us to store the "External system record Id, inside the field". Which allows us to use as the reference during integration.**

**2. Text Area:**

**This DataType allows us to store max. of 255 Characters of content inside the field in multiple lines.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Lenght :**

**6. # Of Lines Visible:**

**7. Required Checkbox:**

**3. Text Area Long:**

**This DataType allows us to enter max. of 1,31,072 Characters of content inside the field.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Length:**

**6. # Of Lines Visible:**

**4. Text Area Rich:**

**This Control allows us to enter max. of 1,31,072 characters of content inside the field, along with the formatting options.**

**i.e. We can change the Color, Font, Style, Size, Alignment, Apply Bullet formats, Include Hyperlinks, Images, Pictures,...etc.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Length:**

**6. # Of Lines Visible:**

**5. Number:**

**This DataType allows us to store max. of an 18-Digit number inside the field including decimal digits.**

**Note: We can specify, the Number of Decimal Digits to be allowed.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Length:**

**6. # Of Decimal Digits:**

**7. Required Checkbox:**

**8. External ID Checkbox:**

**6. Percent:**

**This DataType allows us to store max. of an 18-Digit Number including decimal digits.**

**It will post-fix the number with a "%" symbol automatically.**

**Ex:**

**Discount : 5 ----> Save ---> Discount : 5%**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Length:**

**6. # Of Decimal Digits:**

**7. Required Checkbox:**

**7. Currency:**

**This DataType allows us to store max. of an 18-Digit Number, along with the Decimal Digits upto 2 Digits. It will pre-fix the value with "Currently Configured Currency Symbol" inside the Organization.**

**Note: We can change the Currency Symbol, by using "Company**

**Information" link.**

**Note: We can Activate Multiple Currencies for the Organization**

**by selecting the Checkbox "Activate Multiple Currencies".**

**Ex:**

**Current CTC : $1700000**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Length:**

**6. # Of Decimal Digits:**

**7. Required Checkbox:**

**8. External ID Checkbox:**

**8. Checkbox:**

**Checkbox Allows us to store the Boolean values (i.e. TRUE / FALSE) inside the field.**

**If the Checkbox Selected ---> TRUE**

**If the Checkbox Not Selected ---> FALSE**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Default Option : UnSelected.**

**9. Date:**

**This DataType allows us to store the Date values in (MM/DD/YYYY) format, by selecting from the Pop-up Calendar.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**10. Time:**

**This DataType allows us to store either 12 / 24 Hours format Time values.**

**Ex:**

**HH:MM**

**HH:MM:SS**

**HH:MM:SS:MSec (1 Sec = 1000 Milli Seconds)**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**11. DateTime:**

**This DataType allows us to store both "Date and Time" stamp value inside the field.**

**Ex:**

**CreatedBy, LastModifiedBy, Deletion Date,...etc.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**12. URL:**

**This DataType allows us to store the Website Names / URL / Path of the Applications inside the field.**

**Ex:**

**www.gmail.com**

**https://login.salesforce.com/services/oauth2/callback.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**13. Phone:**

**This DataType allows us to store Phone and Fax values inside the field. It will represent the Value in the form of "US Phone Number" format by default.**

**Ex:**

**Phone 9988776655 ---> Save ---> Phone : (998) 877-6655**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**6. Unique Checkbox:**

**7. External ID Checkbox:**

**14. Email:**

**This DataType allows us to store the Email Id values inside the field. It contains the in-built validation for the email format.**

**Ex:**

**<EmailName>@<DomainName>**

**.com .in .edu .org, .net. .co.in, .au, .us, .gov.in,...etc.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**6. Unique Checkbox:**

**7. External ID Checkbox:**

**15. Picklist:**

**This Controls is used to select only one option from the collection of options.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Values : To specify the List of element names to be get**

**populated inside the control.**

**6. Checkbox: Arrange the Elements in Alphabetical Order.**

**7. Required Checkbox:**

**8. Checkbox: Make the First Element as the Default option.**

**9. Checkbox: Restrict the Picklist values based on the**

**Values Set.**

**16. Picklist-MultiSelect:**

**This DataType allows us to select multiple elements from the collection of elements.**

**Note: Selected Elements will store inside the field, by**

**concatenating all the elements by using a separator "Semicolon".**

**Ex: Football;Reading Books;Music;Movies;Browsing**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Values (LOV's):**

**6. Checkbox : Arrange the Elements in Alphabetical Order.**

**17. Text Encrypted:**

**This DataType is used to Encrypt the specified content into an another format based on the slected "Masked Type".**

**This DataType is used to store the Confidential / Sensitive information like "Passwords, Bank Account Numbers, Credit Card Numbers, US Insurance Policy Numbers, Social Security Numbers,...etc".**

**Ex:**

**Password : \*\*\*\*\*\*\*\*\*\***

**Bank Account : XXX-XXX-5660**

**Credit Card Number : XXXX-XXXX-XXXX-4004**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**

**6. Masked Type:**

**Options: Password, Credit Card Number, US Insurance Policy**

**Number, SSN Numbers,...etc.**

**7. Masked Character: \* X**

**18. Auto Number:**

**Auto Number datatype is used to make the field values to be get "Auto Generated" by Salesforce based on the specified format.**

**While Creating the Auto Number field, we have to specify the format.**

**These fields are also called as "System-Generated" fields. Which can't be Editable. These are "ReadOnly" field.**

**Ex:**

**Products : Product Code(PID-400001) ----> PID-{000000}**

**PID-400002**

**PID-400003**

**....**

**....**

**Customers : Customer Code (C-900001) ---> C-{000000}**

**C-900002**

**C-900003**

**....**

**....**

**Employees : Employee Code: (EID-4001) ----> EID-{0000}**

**EID-4002**

**EID-4003**

**....**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Display Format : (Ex: PID-{000000} )**

**6. Starting Number : 400001**

**7. Checkbox : Generate Auto Number for Existing Records.**

**System-Generated Fields:**

**------------------------**

**Id , CreatedBy, LastModifiedBy, Auto Number, Formula, Roll-up Summary.**

**19. GeoLocation:**

**This DataType allows us to store the Geographical Co-Ordinates of a Location inside the field. i.e. Latitude and Longitude values.**

**Properties:**

**1. Field Label:**

**2. Field Name :**

**3. Description:**

**4. Help Text :**

**5. Required Checkbox:**Again, there are quite a few field types, but most of them are fairly self-explanatory. The important takeaway here is that you want to think about what kind of data you’re trying to store when you create a custom field.

**Create a Custom Field**

The Property object we just created is pretty bare-bones. Let’s add some custom fields to it. Head back to your Trailhead Playground.

# From Setup, go to **Object Manager** | **CustomerPolicyDetails**

1. In the sidebar, click **Fields & Relationships**
2. Click **New** in the top right.
3. For data type, select **Picklist**
4. Click **Next**.
5. Fill out the following:
   1. Field Label: **Policy\_Status\_\_c.**
6. Check the **Required** box.
7. Click **Next**, **Next** again, and then **Save**.

“\_\_c” part is an easy way to tell that a particular field is a custom field.

**Governor Limits:**

**1. In Free Developer Edition:**

An object can have max. of 500 Custom Fields.

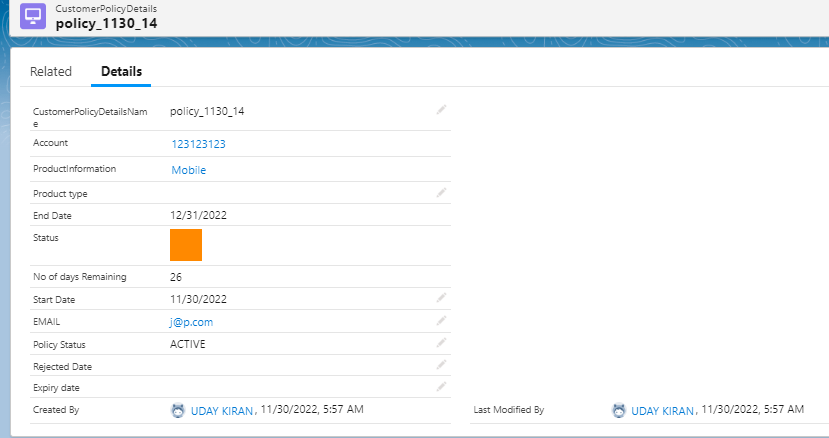
**2. In Unlimited Edition:**

An object can have max. of 800 Custom Fields.

**Create a Record**

Let’s create a property record to see what you did.

1. From the App Launcher (The App Launcher icon. in the navigation bar), find and select **NICHEBIT**.
2. Click the **CustomerPolicyDetails** tab in the navigation bar. If you don’t see it, look under the **More** dropdown.
3. Click **New** in the top corner.
4. Enter a name and price for the property and click **Save**.



**Tabs:**

Tab provides a collection of Graphical User Interfaces, to be used to manage the records inside the associated object.

By using Tab, we can perform all the DML Operations inside the object. i.e. We can perform INSERT, UPDATE, DELETE, UNDELETE, UPSERT, MERGE, and View.

Each Object contains its own Tab. Where Object name is in "Singular format", and "Tab Name is in Plural Format".

Object Name Tab Name

-----------------------------------------

Campaign ----> Campaigns

Lead ----> Leads

Account ----> Accounts

Contact ----> Contacts

Opportunity ----> Opportunities

Case ----> Cases

Solution ----> Solutions

Order ----> Orders

**Ways to Listout the Objects:**

Salesforce provides the below 3 ways to view the objects available in the Organization.

1. **By using "Object Manager**":

Object Manager is a feature in Lightning experience, to be used to manage the Salesforce Objects". i.e. We can View the Objects, Create New Custom Objects, We can Customize / Remove the Objects.

Click on "Setup" menu.

1. Click on "Object Manager" Tab.

2. View the Salesforce Objects.

**2. By using "Developer Console":**

Click on "Setup" menu.

1. Click on "Developer Console" menu item.

2. Goto the Developer Console Editor.

3. Click on "File ---> Open" menu item.

4. Select the Entity Type as "Objects" from Left Panel.

5. View the Objects information in right panel.

6. Double Click on the Object Name, to View the Fields inside the

Object.

3. By using "Schema Builder Tool".

Click on "Setup" menu.

1. Search for the option "Schema Builder" in QuickFind box.

2. Select the "Object's Checkbox" from left panel.

3. View the selected objects on the Canvas.

# **Object Relationships**

Relationships:

==============

While Storing the Whole Business Data inside a Single Table / Object, it causes so many issues as below.

1. It causes the Number of Columns Limitation inside the Object.

2. It causes the Redundancy Issues.

(i.e. In few columns, Duplicate data will get stored)

3. It causes the Performance Issues.

4. Mangaging the Data inside the Table will be very difficuilt.

To avoid the above issues, we have to use a feature called as "Relationships".

i.e. Instead of Storing the Whole Data inside a Single Table, we can split the records to various separate Tables. So that we can manage the records easily.

While mapping the Association / Relationship between the objects, we have to follow the below 2 steps.

Step 1: Identify the Parent Object and Child Object.

Step 2: We have to Map the Relationship between the Objects.

i.e. We have to Create a Relationship DataType field inside the Child Object, which references to the Parent Object.

Note: Salesforce provides the below Relationship DataTypes.

1. Lookup Relationship

2. Master-Detail Relationship.

3. Hierarchical Relationship

4. External Lookup Relationship.

Agenda:

-------

1. Lookup Relationship.

2. Lookup Dialog Configuration.

3. Lookup Filter Configuration.

4. Master-Detail Relationship.

5. Convert Lookup To Master-Detail.

6. Convert Master-Detail To Lookup.

7. Many - Many Relationship

8. Standard Relationships.

9. Standard Junction Objects.

10. Roll-up Summary Field.

11. Hierarchical Relationship

12. One-One Relationship.

**Lookup Relationship:**

Configure the Lookup Relationship between "Hiring Manager" and "Position" objects, and list out all the Observations / Features.

Step 1: Identify the Parent Object and Child Objects.

Parent Object -----> Hiring Manager

Child Object -----> Position

Step 2: Map the Relationship between the Objects.

We have to Create a Field inside the "Position Object" with "Lookup Relationship" datatype, which references to "Hiring Manager Object" as the Parent.

Observations:

1. Lookup Relationship provides One-Many Relationship between the

objects by default.

i.e. A Parent Record can be associated with one or more Child records inside the Child Object.

\*\*\* 2. Lookup Relationship can be applicable on the objects, eventhough both

the objects contains the pre-existing records.

3. In Lookup Relationship, Child Record may / may not be associated with

the "Parent Record".

i.e. While Creating the Child Record, Lookup Field is an optional field.

\*\*\* 4. Re-Parenting option is available by default.

i.e. We can Change the Parent of a Child Record to another Parent at any point of time.

\*\*\* 5. While removing the Parent Record, Only Parent Record will get removed from the

Object. Child Record will get existed as it is.

Note: It will clear the "Lookup Field Value" inside the Child record automatically.

\*\*\* 6. We can make the Lookup Field Required by selecting "Required Checkbox", upon

Creating / Editing the Lookup Field.

\*\*\* 7. If the "Lookup field is Required". Then if the user tries to remove a Parent

Record, which is associated with few child records. Then Salesforce won't allow to remove the Parent Record from the object.

Sol 1: We have to remove all the related Child records from the Child

Object. And then Remove the Parent Record.

Sol 2: We have to re-parent the Child records to some other Parent Record.

And then we can remove the Old Parent from the object.

\*\*\* 8. An object can have max. of 40 Lookup Relationship fields.

i.e. A Child Object can be associated with max. of 40 Parent Objects.

\*\*\* 9. Sharing Settings and Security Settings of the Child Record will be Independent of

the Parent Record.

\*\*\* 10. Roll-up Summary fields can't be applicable on the Lookup Relationship objects.

\*\*\* 11. In Lookup Relationship, Standard object can be a "Child Object" with our Custom

Object as Parent.

Note: i.e. It will support all the Combinations of Objects, while mapping the relation Like Standard-Custom and Vice-versa, and Standard-Standard and Custom-Custom.

**Master Detail Relationship**

1. Master-Detail Association provides One-Many Relationships between the

Objects. Where each Parent Record can be associated with One or More Child records inside the Child object.

2. Master-Detail Relationship can't be applicable on the Child Object, if it

having Pre-existing records.

Sol 1: We have to remove all the Child records from the Child Object

Permanently, and then we can map Master-Detail Relationship.

Sol 2:

1. We have to Create Lookup Relationship between the objects first.

2. And then map each Child Record to the associated Parent Record. 3. Once all the Child Records are associated with the respective

parents, then we can Change the Relationship field datatype from "Lookup to Master-Detail".

3. While Creating a Child Record, Lookup field is Required. i.e. Each Child

Record should be associated with a Parent Record.

4. Re-Parenting option is not available by default. We have to make it enable

manually by selecting the Checkbox "Allow Re-Parenting" while creating / editing the relationship field.

5. While removing the Parent Record, all the associated Child records will get

removed automatically.

6. Sharing Settings and Security Settings of the Child record is purely

dependent on the Parent Record.

7. Roll-up Summary fields can be applicable on the Master-Detail Relationship objects.

8. We can't make a Standard Object as the Child with Master-Detail

Relationship. i.e. Standard Object should be always a Parent object.

9. An object can have max. of 2 Master-Detail Relationship fields.

**Hierarchical:**

Hierarchical Association allows us to map the relation of the object with itself. i.e. It will look like as "Self-Joining".

Note: Hierarchical Association can be applicable only for the "User" object.

i.e. We can map the Relation of a User, with another User based on the Role / Designation.

Ex: Manager and Sub-Ordinates.

**Standard Relationships:**

Salesforce provides a Set of Readymade objects as part of Salesforce CRM. Which can be used to store the Sales, Service and Marketing Related Data. Which are called as "Standard Objects".

Salesforce provides the association among those Standard Objects, to make the records to be get related with each other.

1. Account and Contact:

Between Account and Contact Objects we have "Lookup Relationship" by default. Where

Account Object ----> Parent Object

Contact Object ----> Child Object

Relationship Field ----> Contact:AccountId

(Which Contains the Account Record ID, to which the Contact gets associated)

Note:

1. While Removing the Account Record, all the Related Contact Records

will get removed from the object automatically. It is an Additional behavior given by Salesforce based on "CRM Users Request".

2. Account and Opportunity:

Between Account and Opportunity, we have Lookup Relationship. Where

Account Object -----> Parent Object

Opportunity Object -----> Child Object.

Relationship Field -----> Opportunity:AccountId (Which Contains the Account Record Id, to Which the Opportunity gets associated)

Note:

1. While Removing the Account Record, all the associated

Opportunity Records will get removed automatically.

2. We can Create Roll-up Summary fields between "Account and

Opportunity".

3. Account and Case:

Between Account and Case, we have Lookup Relationship. Where

Account Object -----> Parent Object

Case Object -----> Child Object

Relationship Field -----> Case:AccountId (Which Contains

the Account Record Id, to which the Case Record is Associated)

Note:

While Removing the Account Record, which is associated with the Cases. Then Salesforce won't allow to remove the Account from the object.

Object relationships are a **special field type** that connects two objects together.

There are two main types of object relationships: lookup and master-detail.

**Lookup Relationships**

A lookup relationship essentially links two objects together so that you can “look up” one object from the related items on another object.

Lookup relationships can be one-to-one or one-to-many. The Account to Contact relationship is one-to-many because a single account can have many related contacts

**Master-Detail Relationships**

While lookup relationships are fairly casual, **master-detail relationships** are a bit tighter. In this type of relationship, one object is the master and another is the detail. The master object controls certain behaviors of the detail object, like who can view the detail’s data.

 you use lookup relationships when objects are only related in some cases. Sometimes a contact is associated with a specific account, but sometimes it’s just a contact. Objects in lookup relationships usually work as stand-alone objects and have their own tabs in the user interface.

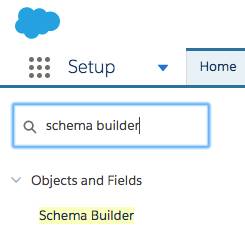
In a master-detail relationship, the detail object doesn’t work as a stand-alone. It’s highly dependent on the master. In fact, if a record on the master object is deleted, all its related detail records are deleted as well. When you’re creating master-detail relationships, you always create the relationship field on the detail object

Finally, you could run into a third relationship type called a hierarchical relationship. Hierarchical relationships are a special type of lookup relationship. The main difference between the two is that hierarchical relationships are only available on the User object. You can use them for things like creating management chains between users.

When you start adding relationships between objects, remember that you’re increasing the complexity of your data model. That’s not a bad thing, but be extra cautious when you do things like change and delete objects, records, or fields. Check out the resources section for more information on relationship behaviors.

Schema Builder is a tool that lets you visualize and edit your data model. It’s useful for designing and understanding complex data models

**Schema Builder:**

1. From Setup, search for and click Schema Builder in the Quick Find box. 
2. In the left panel, click **Clear All**.
3. Check Contact, Favorite, Offer, and Property. You should have the Favorite object from the previous unit, and the Offer and Property objects from the previous challenges.
4. Click **Auto-Layout**.

**Create an Object with Schema Builder**

You can also create objects using Schema Builder. If you prefer, you can create objects in this visual interface if you’re designing your system and want to be able to revise all your customizations on the spot. Let’s see how it’s done.

1. In the left sidebar, click the **Elements** tab.
2. Click **Object** and drag it onto the canvas.
3. Enter information about your object. You can make it whatever you want!
4. Click **Save**.

Your new object appears in the Schema Builder

**Create Fields with Schema Builder**

Creating fields with Schema Builder is just like creating objects.

1. From the Elements tab, choose a field type and drag it onto the object you just created. Notice that you can create relationship fields, formula fields, and normal fields in Schema Builder.
2. Fill out the details about your new field.
3. Click **Save**.

Schema Builder lets you add the following to your schema:

* Custom objects
* Lookup relationships
* Master-detail relationships
* All custom fields except: Geolocation

Schema Builder is a Graphical Representational Tool, Which is used to perform the below 2 Operations.

**1. We can View the Existing Schema:**

By using Schema Builder, We can represent the existing object structure on the Canvas. And we can identify the Relations among the multiple objects.

**2. We can Build our own Schema based on the need**.

By using Schema Builder, we can Create our Own Custom Objects, and We can Add the Fields to the Objects, and We can map the Relations among the objects by using Drag-and-Drop features.

Click on "Setup" menu.

1. Search for the option "Schema Builder".

2. View the Schema Builder Tool.

**Drawbacks:**

While Creating Objects, through Schema Builder we have the below 2 Limitations.

1. Schema Builder will Create a Custom Object, but it won't provide

the Tab for the object.

Sol: We have to Create the Tab for the Custom Object manually

by using "Custom Object Tabs" option.

2. While Creating the Custom Fields to the Object. Schema Builder will represent only the Mandatory fields on the Page Layout by default. Optional fields will not be visible on the Page Layout.

Sol: We have to add the Optional fields onto the Page Layout

manually by using "Page Layout Customizations".