

Create a salesforce custom app on Recruit app with at least 2 objects(Recruiter/Manager, Recruitee/Candidate).Each object should have at least 4 fields with required validation wherever applicable and each object should hold at least 5 records in it. Relate the objects with Master relationship.Also create trigger to assign candidates automatically under a particular Recruiter/Specialization/Domain after insertion.

Create an application having the following custom objects

recruiter :

The screenshot shows the Salesforce Object Manager interface for the 'Recruiter' object. On the left, a sidebar lists various setup options like Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, etc. The main area is titled 'Fields & Relationships' and displays six items, sorted by Field Label. The table has columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The data is as follows:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Domain	Domain_c	Picklist		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
pno	pno_c	Phone		
Recuriter Name	Name	Text(80)		

Add the same domain in the picklist the you are going to add it in job

Ex.

WD

AD

AL

ML

DSA

Job :

The screenshot shows the Salesforce Setup interface with the following details:

- Tab Bar:** LP_2_Chips_(1)(1).pdf, yash | Salesforce, Jobb | Salesforce, Developer Console, New Tab.
- Header:** mmcoe70-dev-ed.lightning.force.com/lightning/setup/ObjectManager/015i0000020W80/FieldsAndRelationships/view
- Page Title:** Jobb
- Left Sidebar:** Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, Scoping Rules.
- Table:** Fields & Relationships (5 items, Sorted by Field Label)

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Description	Description__c	Text Area(255)		
Jobb Name	Name	Text(80)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
- Bottom:** 79°F Clear, Search, Taskbar icons (File, Home, Object Manager), Date/Time (25-05-2023, 01:28), Language (ENG IN).

Candidate :

The screenshot shows the Salesforce Setup interface with the following details:

- Tab Bar:** LP_2_Chips_(1)(1).pdf, yash | Salesforce, Candidate | Salesforce, Developer Console, New Tab.
- Header:** mmcoe70-dev-ed.lightning.force.com/lightning/setup/ObjectManager/015i0000020W6h/FieldsAndRelationships/view
- Page Title:** Candidate
- Left Sidebar:** Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, Scoping Rules.
- Table:** Fields & Relationships (7 items, Sorted by Field Label)

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Candidate Name	Name	Text(80)		
Created By	CreatedById	Lookup(User)		
Email	Email__c	Email		
Jobb	Jobb__c	Master-Detail(Jobb)		
Last Modified By	LastModifiedById	Lookup(User)		
PhoneNo	PhoneNo__c	Phone		
Recruiter name	Recruiter_name__c	Text(30)		
- Bottom:** 79°F Clear, Search, Taskbar icons (File, Home, Object Manager), Date/Time (25-05-2023, 01:27), Language (ENG IN).

Trigger code :

```
trigger CandidateTrigger on Candidate__c(after insert)
{
    // geting job id of candidate
    // here candidate.Job__c is Master relationship
    Id job_id;
    Id candidate_id;
    for (Candidate__c candidate : trigger.new)
    {
        job_id = candidate.Job__c;
        candidate_id = candidate.Id;
    }

    // geting job name using job id
    String job_name;
    for (Job__c job : [ SELECT Id, Name FROM Job__c ])
    {
        if (job.Id == job_id)
        {
            job_name = job.Name;
        }
    }

    List<Candidate__c> CandidateToUpdate = new List<Candidate__c>();

    // now assigning the name of recruiter to the candidate
    // by cheacking is the recruiter has same domain as the candidates job
    for (Recruiter__c recruiter : [ SELECT Id, Name, Domain__c FROM
Recruiter__c ])
    {
        if (recruiter.Domain__c == job_name)
        {

            // if we found the recruiter with ame domain as the candidates
            job then
                // we assign his name to name of Recruiter filed of candidates
        }
    }
}
```

```

        for (Candidate__c cand : [ SELECT Id, Recruiter_name__c FROM
Candidate__c ])
{
    if (cand.Id == candidate_id)
    {
        cand.Recruiter_name__c = recruiter.Name;
        CandidateToUpdate.add(cand);
    }
}
Update CandidateToUpdate;
}

```

Now add job and recruiters :

The screenshot shows a web browser window with the following details:

- Address Bar:** mmoe70-dev-ed.lightning.force.com/lightning/o/Jobb_c/list?filterName=Recent
- Header:** Shows tabs for Chats, Recently Viewed, Candidate, and Developer Console.
- Toolbar:** Includes icons for search, refresh, and other browser functions.
- Content Area:**
 - Jobb:** A navigation bar with links to Recruit, Candidates, Recruiters, and Jobb.
 - Recently Viewed:** A list of 6 items:
 - 1. AI
 - 2. WD
 - 3. AD
 - 4. ML
 - 5. DA
 - 6. DSA
 - Search Bar:** A search bar with placeholder text "Search this list..." and various filter and sort options.
- Bottom:** A taskbar showing the Windows Start button, a search bar, and several pinned application icons (including Microsoft Edge, Google Chrome, and others). It also displays system status like battery level, network, and date/time (01:26, 25-05-2023).

The screenshot shows a browser window with multiple tabs open, including 'LP_2_Chits_(1)(1).pdf', 'Recently Viewed | Recruiters | Sales', 'Candidate | Salesforce', 'Developer Console', and 'New Tab'. The main content area displays the 'Recruit' interface under the 'Recruiters' tab. A list titled 'Recently Viewed' shows four items: 1. A, 2. B, 3. D, and 4. C. Each item has a checkbox next to it. The top right of the list has buttons for 'New', 'Import', and 'Change Owner'. Below the list is a search bar with the placeholder 'Search this list...'. The bottom of the screen shows a Windows taskbar with various icons and system status.

Now at the time of adding candidate keep the recruiter name field empty :

The screenshot shows a browser window with multiple tabs open, including 'LP_2_Chits_(1)(1).pdf', 'New Candidate | Salesforce', 'Candidate | Salesforce', 'Developer Console', and 'New Tab'. The main content area displays the 'Recruit' interface under the 'Candidates' tab. A modal dialog box titled 'New Candidate' is open, showing the 'Information' section. The 'Candidate Name' field contains 'yash', 'PhoneNo' contains '1234567890', and 'Jobb' contains 'AI'. The 'Recruiter name' field is empty. Below the form are buttons for 'Cancel', 'Save & New', and 'Save'. The background shows the 'Recently Viewed' list of recruiters. The bottom of the screen shows a Windows taskbar with various icons and system status.

It will get added automatically :

The screenshot shows a Microsoft Edge browser window with several tabs open. The active tab is 'Candidate | Salesforce' at the URL mmcoe70-dev-ed.lightning.force.com/lightning/r/Candidate__c/a05i00000GjzN8AAJ/view. A green success message box is displayed in the center of the page, stating 'Candidate **yash** was created.' Below the message, the candidate's details are listed in a table:

Related	Details
Candidate Name	yash
PhoneNo	1234567890
Jobb	Ai
Recruiter name	B
Email	yj@gmail.com

At the bottom of the page, it shows 'Created By' and 'Last Modified By' both as 'Yash Jadhav, 25/05/2023, 1:26 am'. The browser taskbar at the bottom includes icons for weather (79°F), search, file explorer, and various applications like Google Chrome, Microsoft Edge, and Microsoft Word.

Congratulations on wasting your life in engineering 😊

Create a salesforce custom app on Project management app with at least 2-3 objects. Each object should have at least 4 fields with required validation wherever applicable and each object should hold at least 5 records in it. Relate the objects with Master relationship. Also create trigger to assign projects automatically under a particular team after insertion.

Create an application having the following custom objects

domain :

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for 'Recently Viewed', 'Domain | Salesforce', 'Developer Console', and 'Yash Jadhav CC PR OR + G'. The main page title is 'SETUP > OBJECT MANAGER' and the specific object being viewed is 'Domain'. On the left, a sidebar lists various setup categories like 'Fields & Relationships', 'Page Layouts', 'Lightning Record Pages', etc. The central area displays a table titled 'Fields & Relationships' with the following data:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Description	Description_c	Text Area(255)		
Domain Name	Name	Text(80)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		

team :

Team

Fields & Relationships
6 items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Domain	Domain__c	Master-Detail(Domain)	✓	▼
Head	Head__c	Text(30)		▼
Last Modified By	LastModifiedById	Lookup(User)		▼
No of team members	No_of_team_members__c	Number(18, 0)		▼
Team Name	Name	Text(80)	✓	▼

Project :

Project

Fields & Relationships
7 items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Description	Description__c	Text Area(255)		▼
Domain	Domain__c	Master-Detail(Domain)	✓	▼
End Date	End_Date__c	Date		▼
Last Modified By	LastModifiedById	Lookup(User)		▼
Project Name	Name	Text(80)	✓	▼
Team name	Team_name__c	Text(30)		▼

Trigger code :

```
trigger ProjectTrigger on Project__c(after insert)
{

    // geting project id of candidate
    // here project.Domain__c is Master relationship
    Id domain_id;
    Id project_id;
    for (Project__c project : trigger.new)
    {
        domain_id = project.Domain__c;
        project_id = project.Id;
    }

    List<Project__c> projectToUpdate = new List<Project__c>();
    // now assigning the team to the project
    // by cheacking is the team has same domain as the project
    for (Team__c team : [ SELECT Id, Name, Domain__c FROM Team__c ])
    {
        if (team.Domain__c == domain_id)

        {
            // if we found the team with same domain as the project then
            // we assign teams name to name of team in project
            for (Project__c project : [ SELECT Id, Team_name__c FROM
Project__c ])
            {
                if (project.Id == project_id)
                {
                    project.Team_name__c = team.Name;
                    projectToUpdate.add(project);
                }
            }
        }
    }
    Update projectToUpdate;
}
```

Now add domain and team :

Domains

Recently Viewed

5 items • Updated a few seconds ago

Domain Name
1 DA
2 ML
3 WD
4 AD
5 AI

Teams

Recently Viewed

4 items • Updated a few seconds ago

Team Name
1 T4
2 T3
3 T2
4 T1

Now at the time of adding project keep the team name field empty :

The screenshot shows the Salesforce Project Management interface. A 'New Project' dialog box is open, displaying fields for 'Project Name' (chatGPT), 'Description' (the god), 'End Date' (27/05/2023), 'Team name' (empty), and 'Domain' (AI). The 'Team name' field is highlighted with a yellow background.

It will get added automatically :

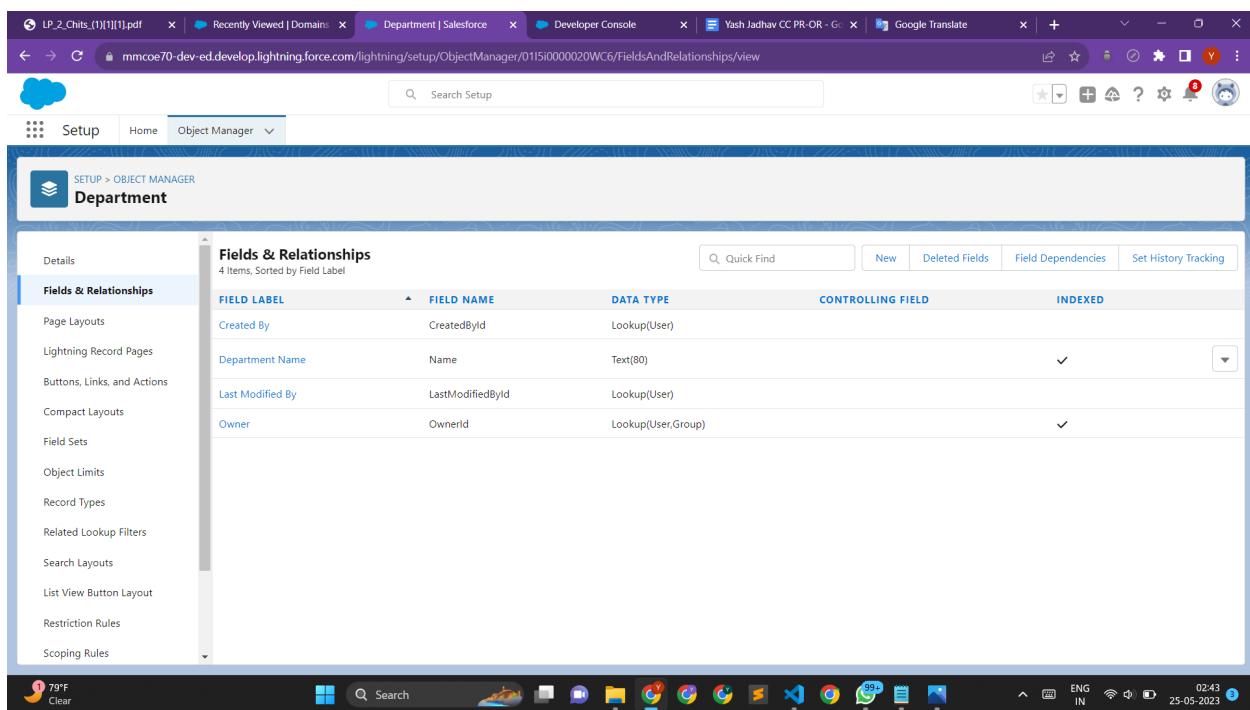
The screenshot shows the details of a newly created project named 'chatGPT'. The 'Team name' field is populated with 'T1'. A green success message at the top of the page states 'Project [chatGPT] was created.'

Congratulations on wasting your life in engineering 😊

Create a salesforce custom app on the School database with at least 2-3 objects(Teacher, student,Batch).Each object should have at least 4 fields with required validation wherever applicable and each object should hold at least 5 records in it. Relate the objects with Master relationship.Also create trigger to assign students automatically under a particular batch after insertion

Create an application having the following custom objects

Department :



The screenshot shows the Salesforce Object Manager interface. The URL in the browser is mmcoe70-dev-ed.lightning.force.com/lightning/setup/ObjectManager/0115i0000020WC6/FieldsAndRelationships/view. The page title is "Department". On the left, there's a sidebar with links like Setup, Home, and Object Manager. The main content area is titled "Fields & Relationships" and lists four items: Created By, Department Name, Last Modified By, and Owner. Each item has a "FIELD LABEL", "FIELD NAME", "DATA TYPE", "CONTROLLING FIELD", and "INDEXED" column. The "Department Name" field is highlighted with a red border.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Department Name	Name	Text(80)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓

Batch :

The screenshot shows the Salesforce Setup interface for the 'Batch' object. On the left, a sidebar lists various setup categories like Page Layouts, Lightning Record Pages, and Field Sets. The main content area is titled 'Fields & Relationships' and displays a table with the following data:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Batch Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
Department	Department__c	Master-Detail(Department)		✓
From	From__c	Number(18, 0)		
Last Modified By	LastModifiedById	Lookup(User)		
To	To__c	Number(18, 0)		
Year	Year__c	Number(18, 0)		

Students :

The screenshot shows the Salesforce Setup interface for the 'Student' object. On the left, a sidebar lists various setup categories. The main content area is titled 'Fields & Relationships' and displays a table with the following data:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Batch	Batch__c	Text(30)		
Created By	CreatedById	Lookup(User)		
Department	Department__c	Master-Detail(Department)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Roll no	Roll_no__c	Number(18, 0)		
student Name	Name	Text(80)		✓
Year	Year__c	Number(18, 0)		

Trigger code :

```
trigger StudentTrigger on Stuudent__c(after insert)
{

    // geting project id of candidate
    // here project.Domain__c is Master relationship
    Id department_id;
    Id student_id;
    Decimal year;
    Decimal rno;
    for (Stuudent__c stud : trigger.new)
    {
        student_id = stud.Id;
        department_id = stud.Department__c;
        year = stud.Year__c;
        rno = stud.Roll_no__c;
    }

    List<Stuudent__c> studentToUpdate = new List<Stuudent__c>();
    // now assigning the team to the project
    // by cheacking is the team has same domain as the project
    for (Batch__c batch : [ SELECT Id, Name, From__c, To__c, Year__c,
Department__c FROM Batch__c ])
    {
        if (batch.Department__c == department_id && batch.Year__c == year
&& ((rno <= batch.To__c) && (rno >= batch.From__c)))
        {

            // if we found the team with same domain as the project then
            // we assign teams name to name of team in project
            for (Stuudent__c stud : [ SELECT Id, Batch__c FROM Stuudent__c
])
            {
                if (stud.Id == student_id)
                {
                    stud.Batch__c = batch.Name;
                }
            }
        }
    }
}
```

```
        studentToUpdate.add(stud);
    }
}
}

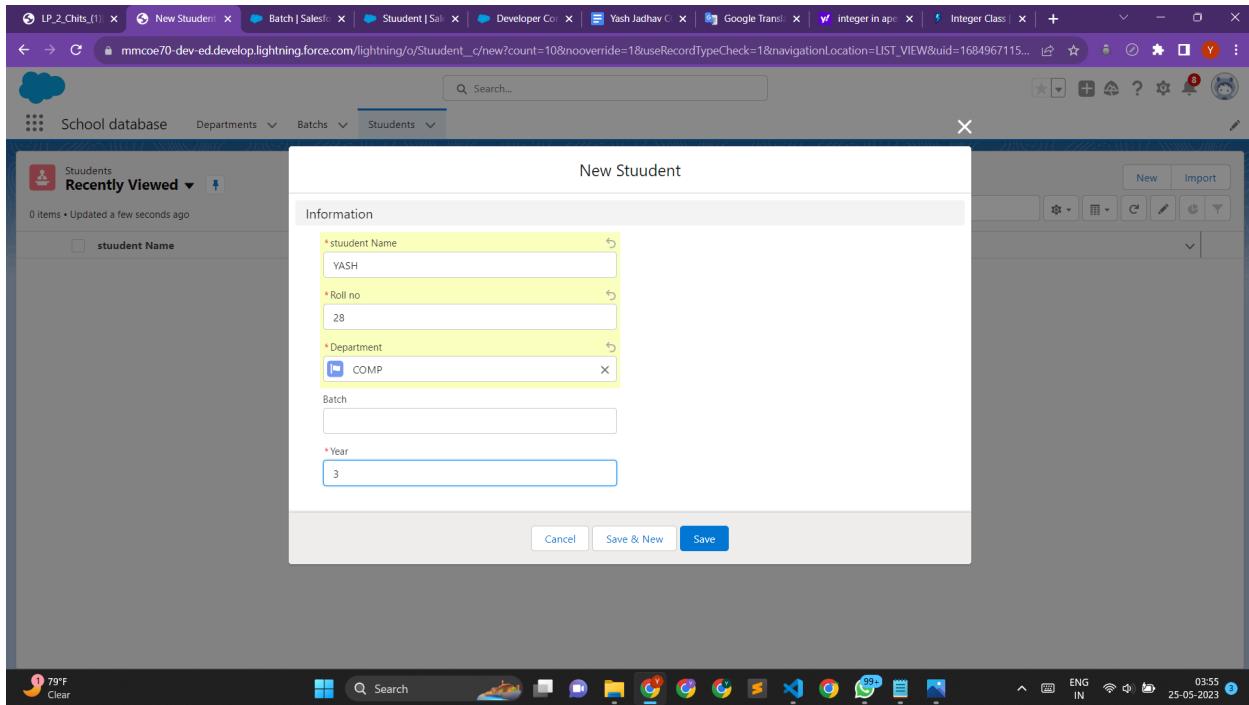
Update studentToUpdate;
}
```

Now add department and batch :

The screenshot shows a web browser window with multiple tabs open at the top. The active tab is 'mmcoe70-dev-ed.lightning.force.com/lightning/o/Department_c/list?filterName=Recent'. The page displays a 'Recently Viewed' list for 'Departments'. The list contains four items, each with a checkbox and a department name: 1. COMP, 2. MECH, 3. ENTC, and 4. IT. At the top right of the list area, there are buttons for 'New', 'Import', and 'Change Owner'. Below the list is a search bar labeled 'Search this list...' and a set of filter and sorting icons.

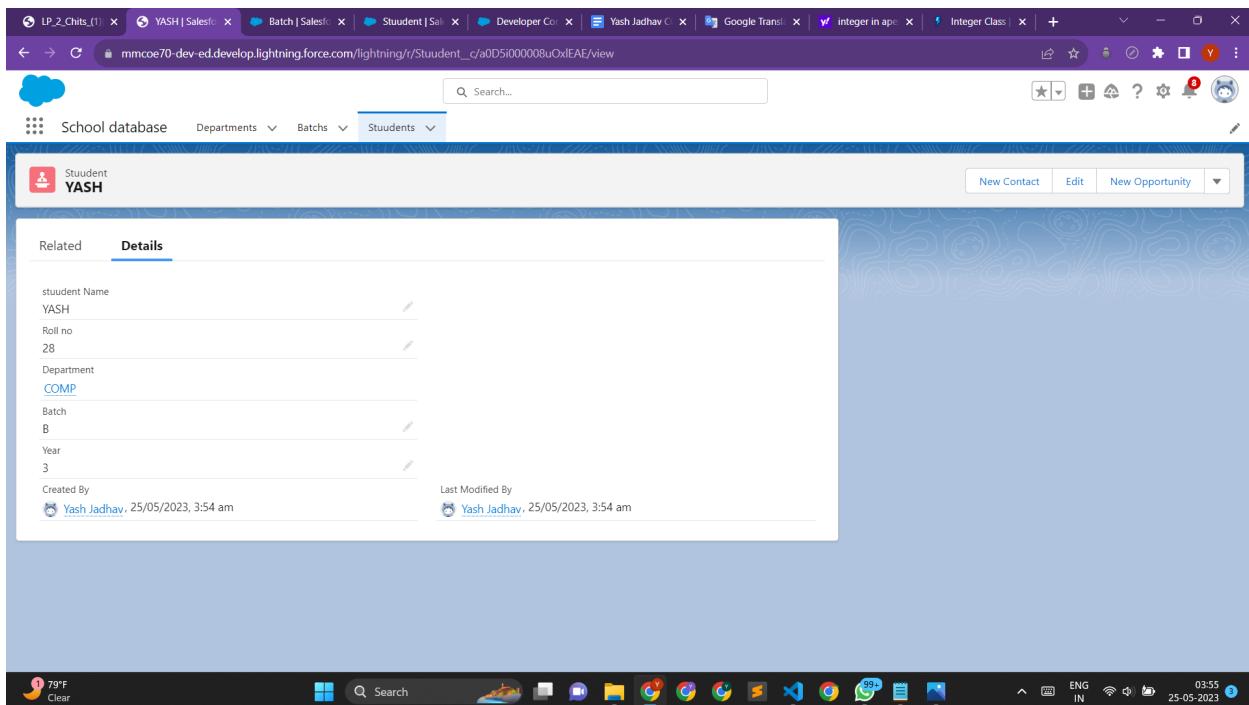
The screenshot shows a web browser window with multiple tabs open at the top. The active tab is 'mmcoe70-dev-ed.lightning.force.com/lightning/o/Batch_c/list?filterName=Recent'. The page displays a 'Recently Viewed' list for 'Batches'. The list contains three items, each with a checkbox and a batch name: 1. C, 2. B, and 3. A. At the top right of the list area, there are buttons for 'New', 'Import', and 'Change Owner'. Below the list is a search bar labeled 'Search this list...' and a set of filter and sorting icons.

Now at the time of adding a student keep the batch field empty :



The screenshot shows the Salesforce Lightning interface for creating a new student. The 'Information' section contains fields for 'student Name' (YASH), 'Roll no' (28), 'Department' (COMP), 'Batch' (empty), and 'Year' (3). The 'Save' button is highlighted.

It will get added automatically :



The screenshot shows the Salesforce Lightning interface displaying the details of a student named YASH. The 'Details' tab is selected, showing fields for student Name (YASH), Roll no (28), Department (COMP), Batch (B), and Year (3). The 'Created By' field shows 'Yash Jadhav, 25/05/2023, 3:54 am'. The 'Last Modified By' field also shows 'Yash Jadhav, 25/05/2023, 3:54 am'.

Congratulations on wasting your life in engineering 😊

Create a salesforce custom app on maharashtra tourism app with at least 2-3 objects. Each object should have at least 4 fields with required validation wherever applicable and each object should hold at least 5 records in it. Relate the objects with the Master relationship. Also create trigger to assign the place automatically under a particular category after insertion

Create an application having the following custom objects

City :

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for 'Setup', 'Home', and 'Object Manager'. The main content area is titled 'City' under 'SETUP > OBJECT MANAGER'. On the left, a sidebar lists various setup options like 'Page Layouts', 'Lightning Record Pages', and 'Field Sets'. The right side displays the 'Fields & Relationships' section for the 'City' object. This section contains a table with columns: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The table rows are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Category	Category__c	Text(30)		
City Name	Name	Text(80)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		

Place :

The screenshot shows the Salesforce Object Manager interface for the 'Place' object. The left sidebar lists various setup options like Page Layouts, Lightning Record Pages, and Field Sets. The main content area displays the 'Fields & Relationships' section, which lists seven fields: Category_of_palce_c, City__c, Created By, Description__c, Last Modified By, Name, and Rating__c. The table includes columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED status.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Category of palce	Category_of_palce_c	Text(30)		
City	City__c	Master-Detail(City)		✓
Created By	CreatedById	Lookup(User)		
Description	Description__c	Text(30)		
Last Modified By	LastModifiedById	Lookup(User)		
Place Name	Name	Text(80)		✓
Rating	Rating__c	Number(18, 0)		

Trigger code :

```
trigger PlaceTrigger on Place__c(after insert)
{
    Id city_id;
    Id place_id;
    for (Place__c place : trigger.new)
    {
        place_id = place.Id;
        city_id = place.City__c;
    }

    String category;
    for (City__c city : [ SELECT Id, Category__c FROM City__c ])
    {
        if (city.Id == city_id)
        {
            category = city.Category__c;
        }
    }

    List<Place__c> placeToUpdate = new List<Place__c>();
    for (Place__c place : [ SELECT Id, Category_of_palce__c FROM Place__c
])
    {
        if (place.Id == place_id)
        {
            place.Category_of_palce__c = category;
            placeToUpdate.add(place);
        }
    }
    Update placeToUpdate;
}
```

Now add cities :

Screenshot of a web browser showing a Salesforce Lightning interface for managing cities. The page displays a 'Recently Viewed' list for the 'cities' category, showing three items: Lonavala, Aurangabad, and Mumbai. The browser tabs include 'LP_2_Chips_(1)(1).pdf', 'Place | Salesforce', 'Recently Viewed | cities | Salesforce', 'Developer Console', and 'list of cities - Yahoo India Search'. The system status bar at the bottom shows '90°F Sunny' and the date '25-05-2023'.

Now at the time of adding a place keep the category field empty :

Screenshot of a web browser showing a Salesforce Lightning interface for creating a new place. The 'New Place' form is open, showing fields for Place Name (Ajanta Caves), City (Aurangabad), Description (Ancient Buddhist caves), and Rating (4.7). The 'Category of place' field is empty. The browser tabs include 'LP_2_Chips_(1)(1).pdf', 'Place | Salesforce', 'New Place | Salesforce', 'Developer Console', and 'list of cities - Yahoo India Search'. The system status bar at the bottom shows '90°F Sunny' and the date '25-05-2023'.

It will get added automatically :

The screenshot shows a Microsoft Edge browser window with multiple tabs open. The active tab is a Lightning component from the 'Place' object in Salesforce. The URL in the address bar is mmcoe70-dev-ed.lightning.force.com/lightning/r/Place_c/a0G5i00000G5xEvEA/view. The page displays a success message: "Place 'Ajanta Caves' was created." Below this, the 'Details' tab is selected, showing the following fields:

Field	Value
Place Name	Ajanta Caves
City	Aurangabad
Description	Ancient Buddhist caves
Rating	5
Category of place	Caves

At the bottom of the form, it says "Created By" followed by a user icon and the name "Yash Jadhav, 25/05/2023, 12:02 pm". The browser's status bar at the bottom shows the date as 25-05-2023 and the time as 12:03.

Congratulations on wasting your life in engineering 😊

For validation of fields

- 1) Go to object manager
- 2) Select the object
- 3) Scroll down on left menu and select “validation rules”
- 4) Click on “new”
- 5) Give the name for rule
- 6) Now write validation rule in “**Error Condition Formula**”

Ex

For phone no:

OR(

 NOT(LEN(PhoneNo__c) = 10),
 ISBLANK(PhoneNo__c),
 NOT(ISNUMBER(PhoneNo__c))

)

For email :

NOT(REGEX(Email__c , '^([a-zA-Z0-9_\\-\\.]+) @ ([a-zA-Z0-9_\\-\\.]+) \\.([a-zA-Z]{2,5}) \$'))

- Change your filed name accordingly

- 7) Write the error message that should be displayed after the wrong value is entered
- 8) Select field and select the particular field where you want to display error message