RFP Demo with Vector Databases and Prompt Builder copie

🏆 Demo Owner

Sharda Rao - Distinguished Technical Architect Data Cloud

🙏 Thank you

Matt Dykeman - Technical Architect Data Cloud
Reinier van Leuken - Product Manager Salesforce
Raveesh Raina - Principal Account SE
Jonathan Zhou - Distinguished Technical Architect

👥 Business Problem

Sales representatives often find themselves grappling with the arduous task of responding to RFP questions, which can be both tedious and time-consuming.

However, a revolutionary solution has emerged with the integration of vector databases and RAG (Retrieval Augmented Generation). This innovation allows sales professionals to leverage their historical RFP responses to automatically generate content for their current inquiries.

This groundbreaking feature is now seamlessly integrated into Salesforce, along with the robust capabilities of a data cloud. Salesforce representatives can now effortlessly generate context-specific content directly within the system of engagement.

Furthermore, they have the flexibility to modify the generated response and update answers accordingly, leading to a significant boost in productivity and an enhanced adoption of Salesforce within the sales team.

By harnessing the power of vector databases and RAG technology, this transformative feature not only streamlines the RFP response process but also empowers sales reps to customize and optimize their answers within the Salesforce ecosystem. This not only saves time but also contributes to an overall increase in efficiency and effectiveness, ultimately driving improved sales performance.

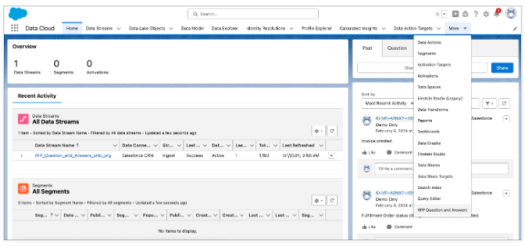
📍 Demo Script

Demo Link - https://clicktologin.herokuapp.com/?un=sharda@rag.demo&pw=salesforce1

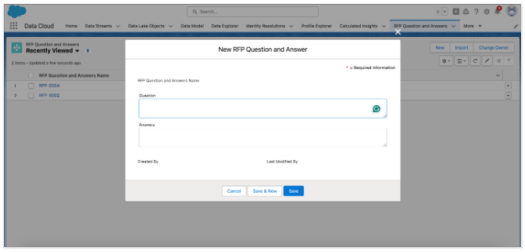
NOTE - The password has been changed. Please DM Sharda Rao for the new password.

Video - https://drive.google.com/file/d/11tVJKl6Uj4aMG3_wt8LTPnh6QP3io1w3/view?usp=sharing

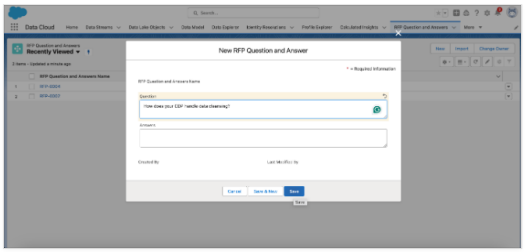
Navigate to Data Cloud → RFP Question and Answers



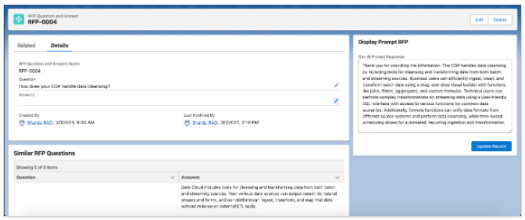
Click New



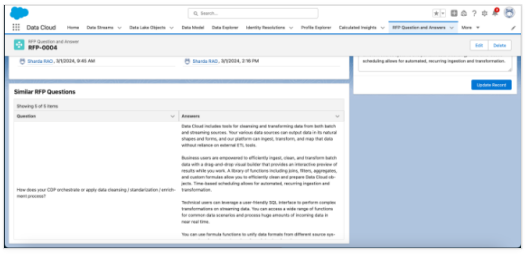
Enter the Question and Click Save



Expand the RFP text box on the right panel. Highlight the text generated by prompt template

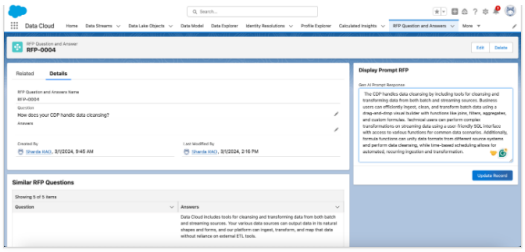


Scroll down and show the similar question and answer generated by vector search

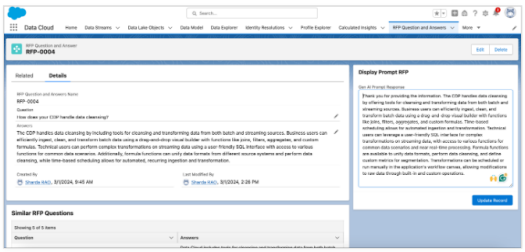


Highlight how you can change the text generated by LLM and click save to update the

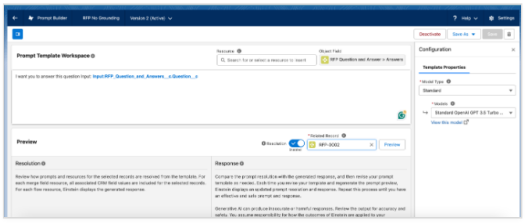
answer in Salesforce object



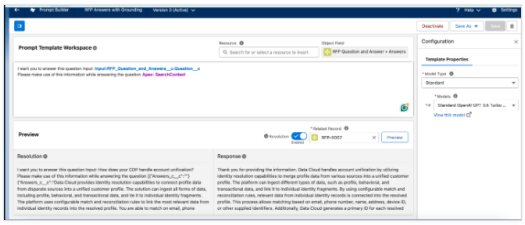
Highlight how the record got updated after Sales rep reviews and updates the answer



Navigate to Set up → Prompt Builder → RFP No Grounding → Choose RFP002 as a related record. Highlight how with no context the LLM response is not relevant for the Sales rep



Navigate to Set up → Prompt Builder → RFP with Grounding → Choose RFP002 as a related record. Highlight how we are now sending the context from our vector search of answers to similar questions on the left panel and show the grounded response on the right panel. Mention how the context is sent using an apex class search context.



In essence, this demonstration effectively illustrates the seamless integration of vector search and the Retrieval Augmented Generator (RAG) within Salesforce DataCloud. The primary goal is to showcase to customers the effortless application of these technologies, specifically emphasizing their ability to anchor prompts within the natural workflow of Salesforce users. By doing so, the demo aims to highlight the enhanced ease with which users can navigate and leverage these tools, ultimately resulting in a significant boost in productivity. This showcase not only underscores the user-friendly nature of the system but also accentuates its capacity to streamline processes, making it an invaluable asset for the Salesforce user community.

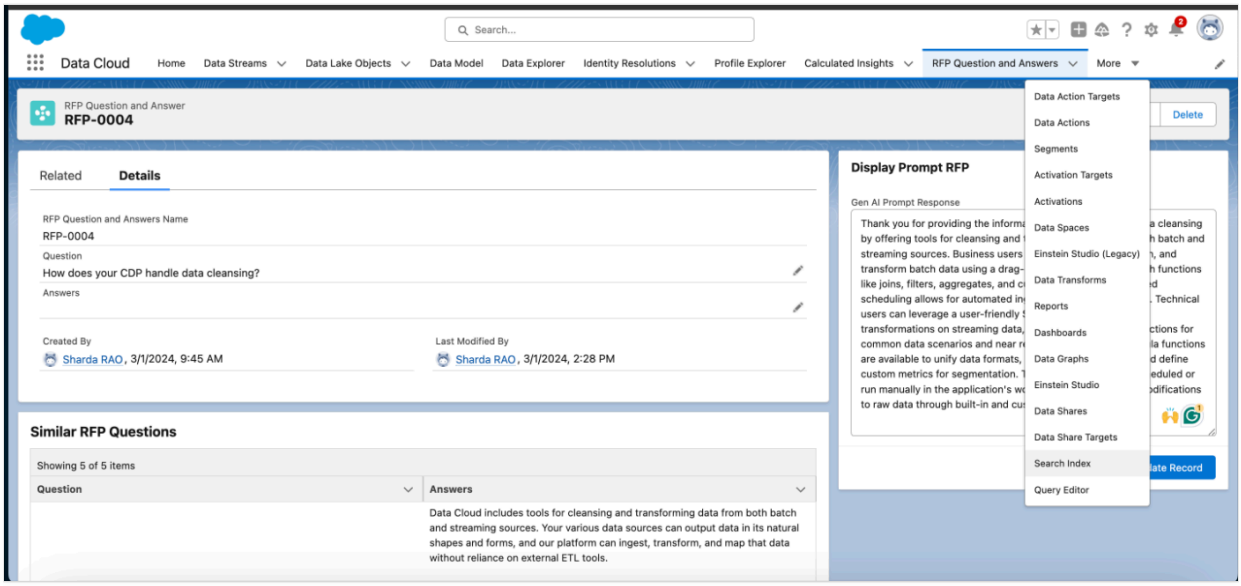
WIP - PDF and knowledge articles vector databases

🛎️ How to build your own

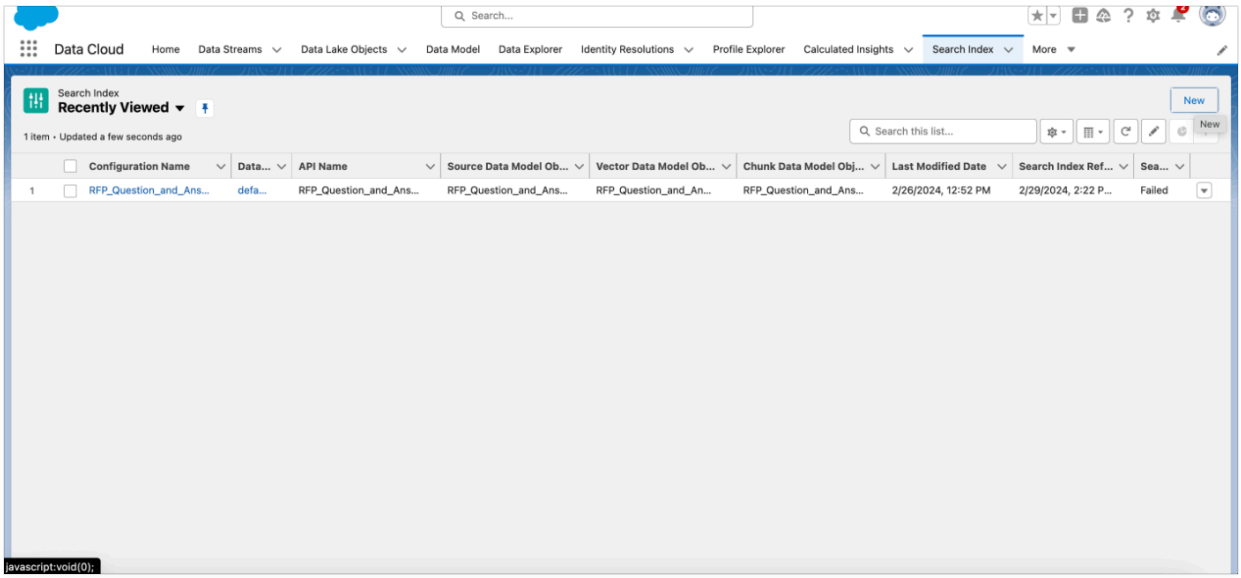
1. Historical dataset to create the custom object. Note that you can use any unstructured data pertinent to your use case

   FP Responses.csv

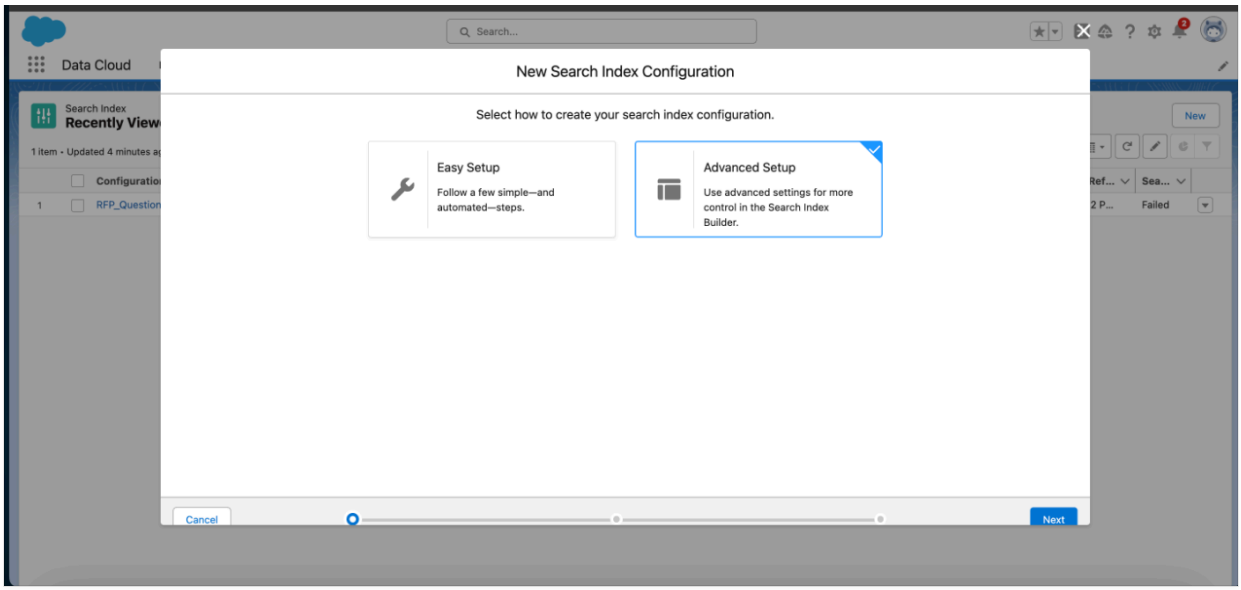
2. Ingest the data into DataCloud as a Data Stream
3. Map it to a custom DMO
4. Navigate to Data Cloud → Search Index



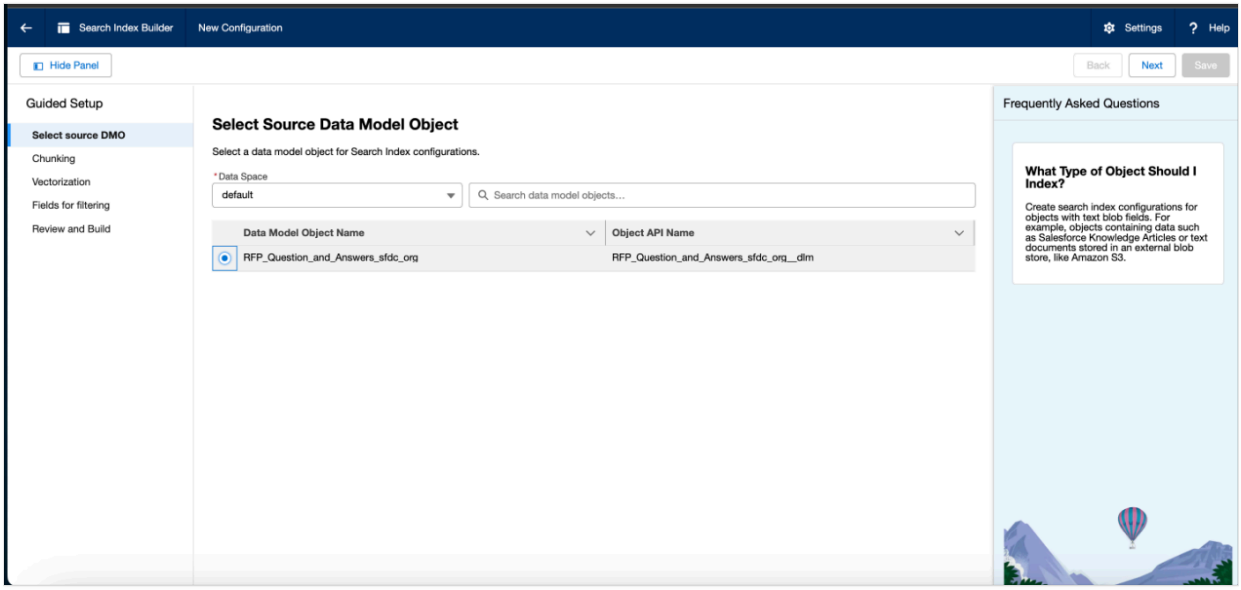
5. Create new



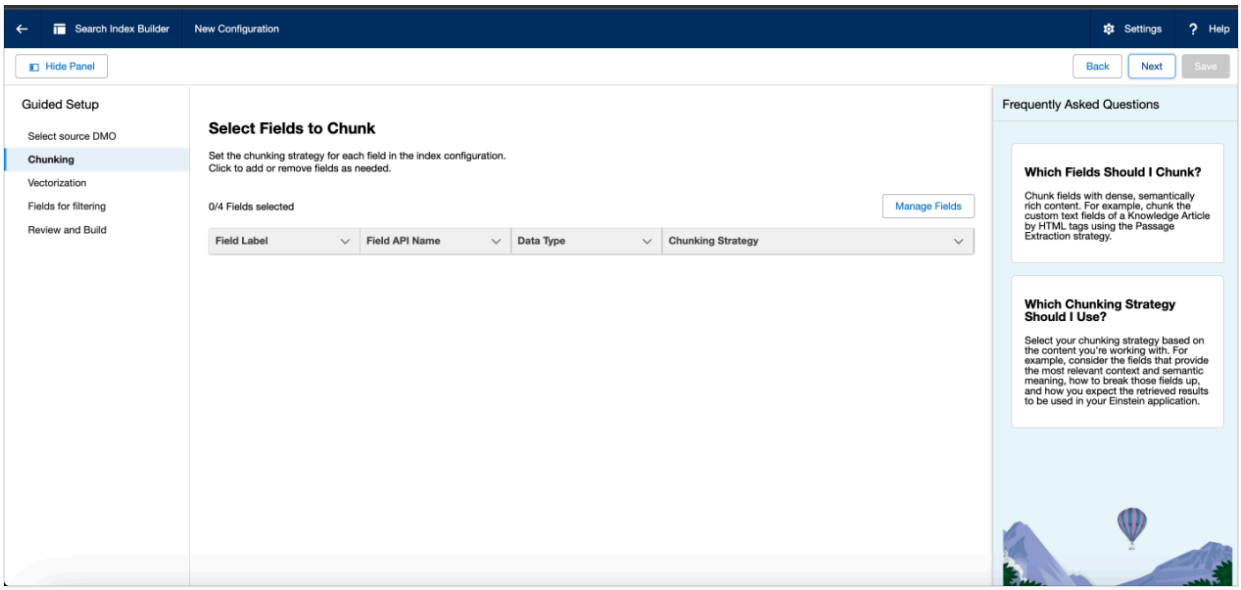
6. Select Advanced and next



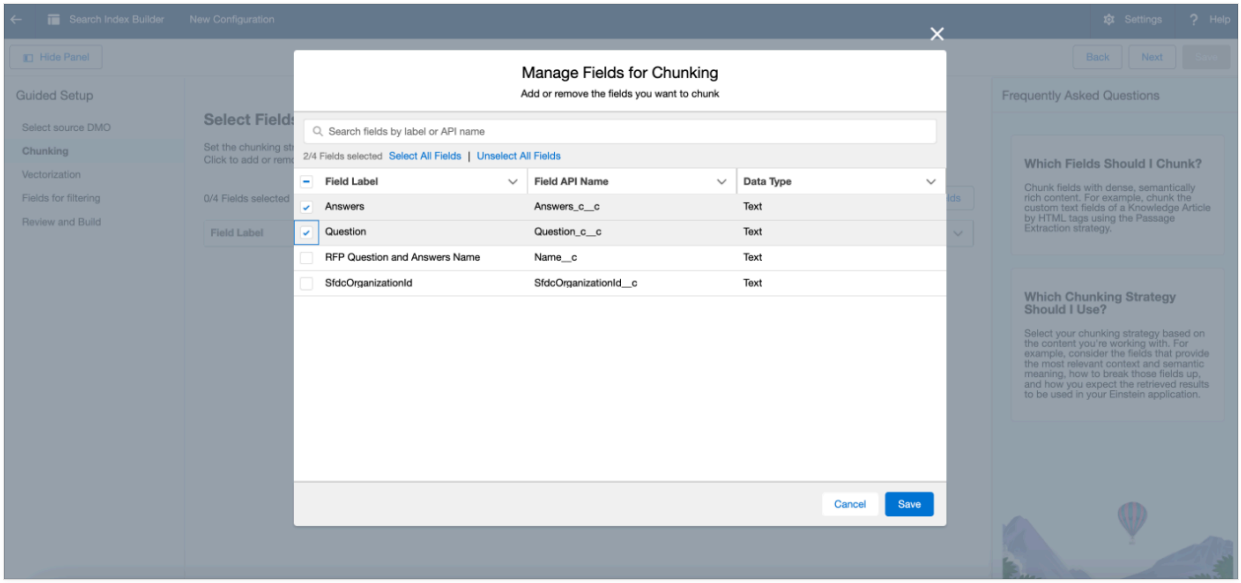
7. Choose DMO



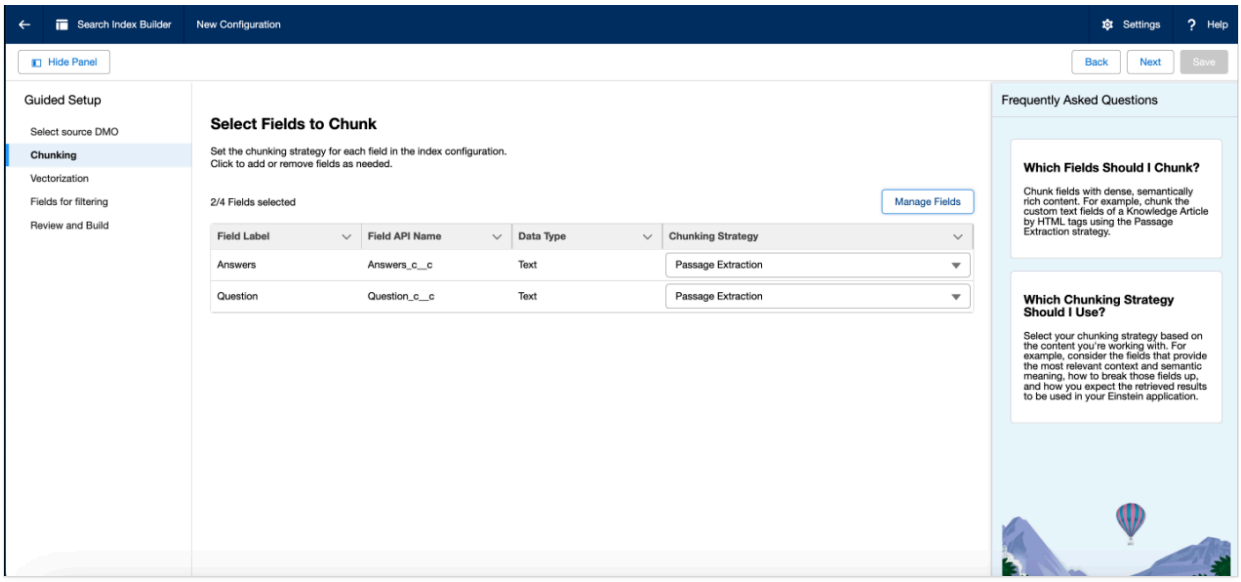
8. Click Next



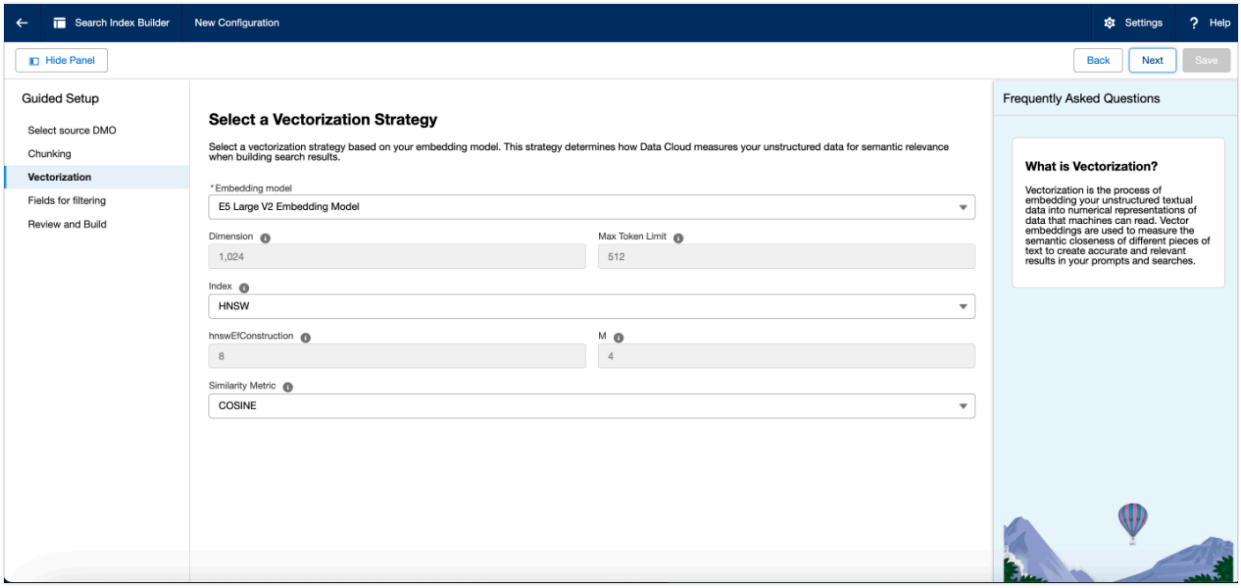
9. Click Manage Fields and select the fields for chunking and click save



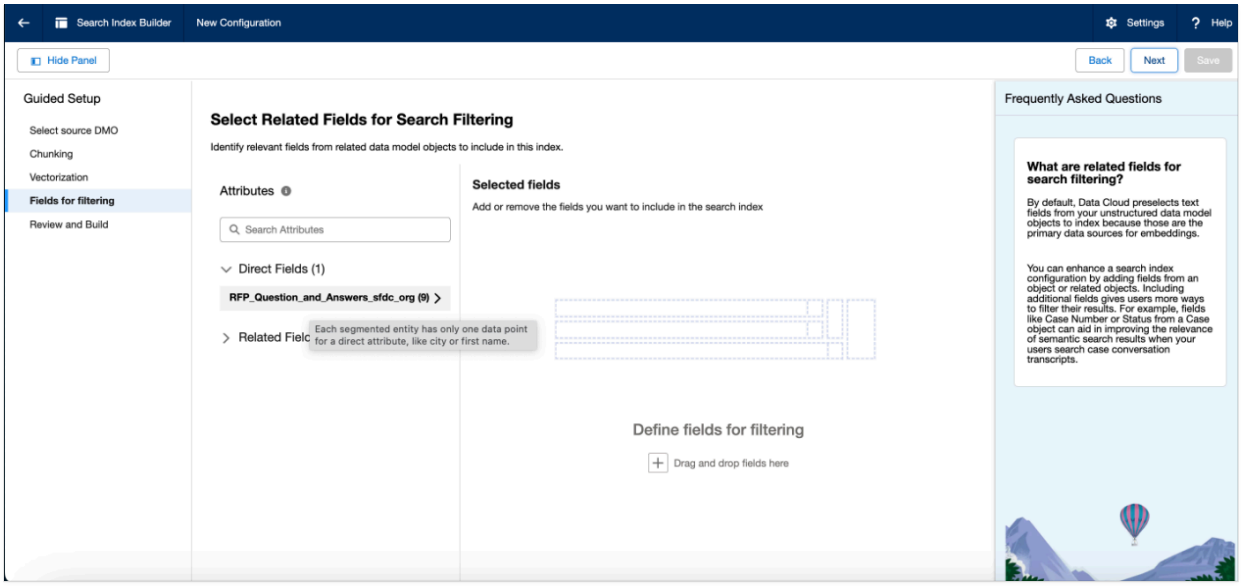
10. Click Next



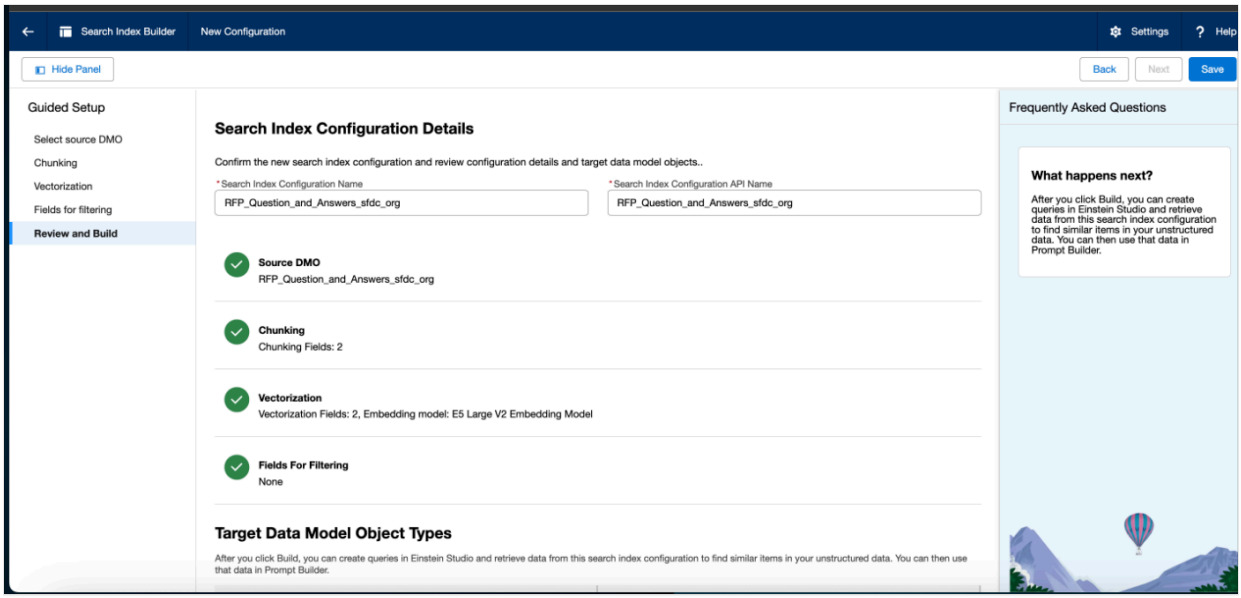
11. Click Next



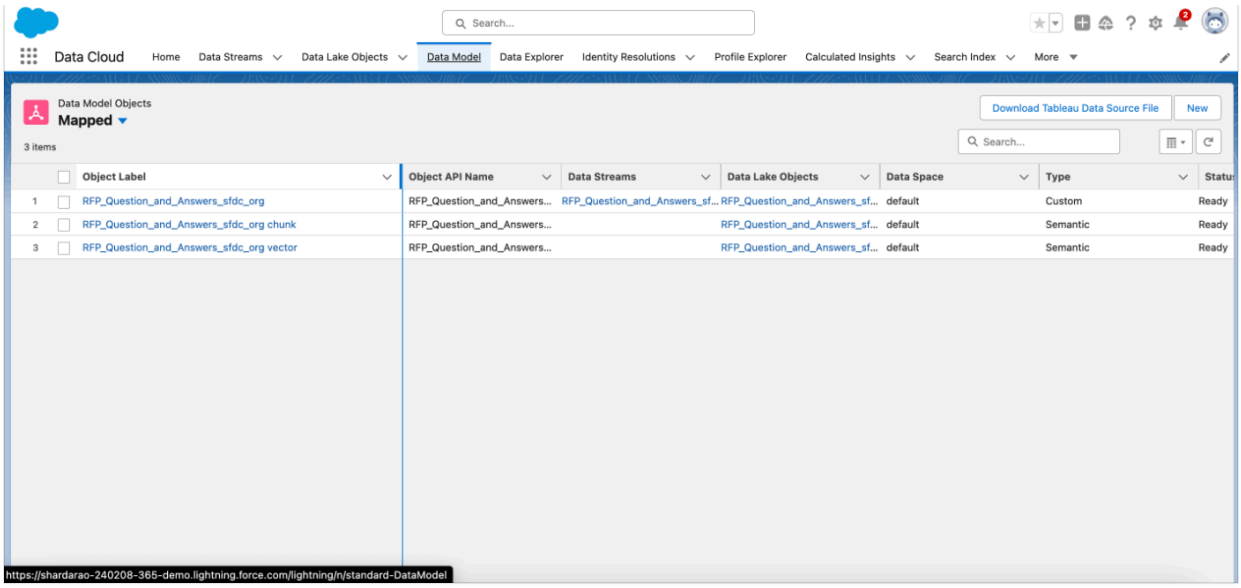
12. If required add filters



13. Review and Save

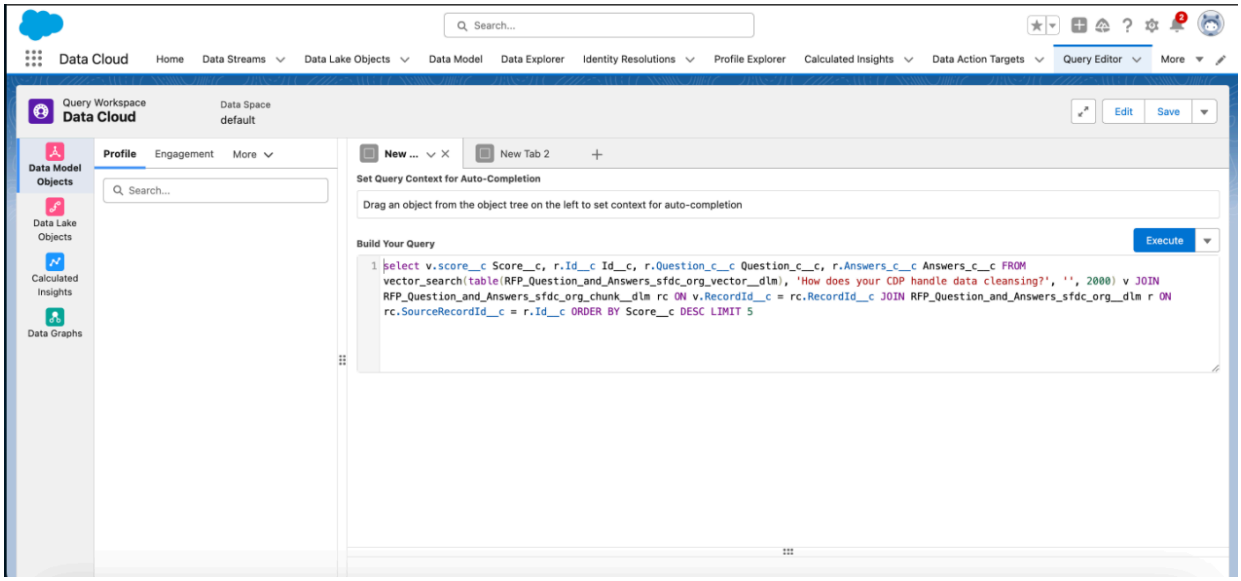


14. Once indexing is completed you will see two new DMOs created



15. Navigate to the query editor and enter the following query

```
select v.score__c Score__c, r.Id__c Id__c, r.Question_c__c Question_c__c, r.Answers_c_
Answers_c__c FROM vector_search(table(RFP_Question_and_Answers_sfdc_org_vector__dlm),
'How does your CDP handle data cleansing?', '', 2000) v
JOIN RFP_Question_and_Answers_sfdc_org_chunk__dlm rc
ON v.RecordId__c = rc.RecordId__c
JOIN RFP_Question_and_Answers_sfdc_org__dlm r ON
rc.SourceRecordId__c = r.Id__c ORDER BY Score__c DESC LIMIT 5
```



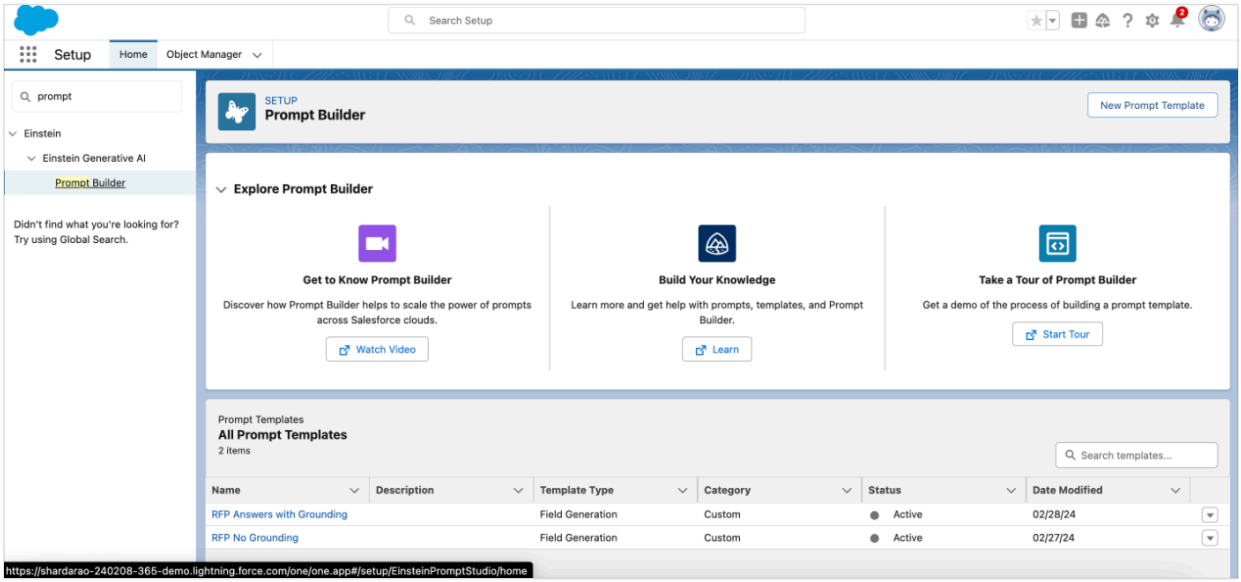
This query fetches all the questions and answers which are very similar to the question 'How does your CDP handle data cleansing?'

16. Create apex class

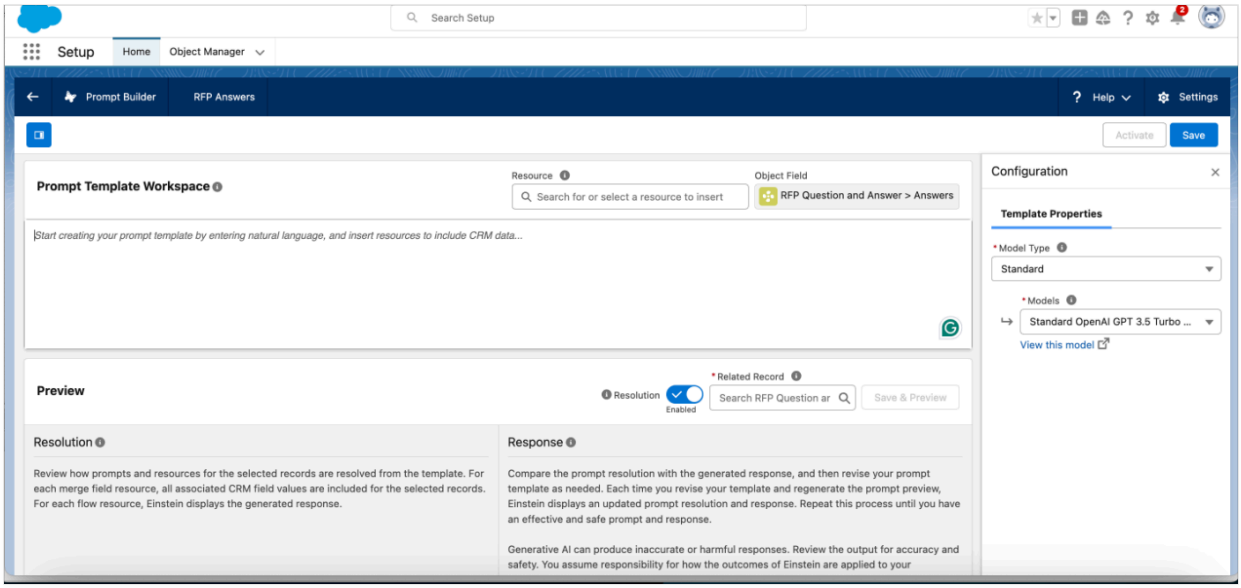
```
public class SearchContext {

    @InvocableMethod(CapabilityType = 'PromptTemplateType://einstein_gpt__fieldComplet
    public static List<Response> searchforContext(List<Request> requests) {
        List<Response> responses = new List<Response>();
        Response response = new Response();

        String topic = requests[0].RelatedEntity.Question__c;

          ConnectApi.CdpQueryInput input = new ConnectApi.CdpQueryInput();
input.sql = 'select v.score__c Score__c, r.Id__c Id__c, r.Answers_c__c Answers_c__c FR

        ConnectApi.CdpQueryOutput output = ConnectApi.CdpQuery.queryANSISql(input);
        List<Object> data = output.data;
        String scs = '';
        List<List<Results>> allResults = new List<List<Results>>();
        List<Results> returnResult = new List<Results>();
        for (Object searchRecord : data) {
            Map<String, Object> myMap = (Map<String, Object>) JSON.deserializeUntyped(
            // check for access of case record for the current user
            if (SearchContext.getUserRecordAccess((String) myMap.get('Id__c'))) {
                Map<String, String> sc = new Map<String, String>();
                //sc.put('Id', (String) myMap.get('Id__c'));
                //sc.put('Similar_Case__c', (String) myMap.get('Id__c'));
                // sc.put('Summary__c', (String) myMap.get('Summary__c'));
                sc.put('Answers_c__c', String.valueOf(myMap.get('Answers_c__c')));
                //sc.put('Summary_c', String.valueOf(myMap.get('Summary__c')));
                scs = scs + JSON.serialize(sc);
            }
        }

        scs = '[' + scs + ']';

        response.Prompt = scs;
        responses.add(response);
        return responses;
    }

    public static boolean getUserRecordAccess(String recordId) {
        return true;
    }

    // All inputs will be passed into the IA
    public class Request {
        @InvocableVariable
        public RFP_Question_and_Answers__c RelatedEntity;
    }

    public class Response {
        @InvocableVariable
        public String Prompt;
    }

    public class Results {
        @InvocableVariable
```

```
        public String Answer;
        @InvocableVariable
        public String Question;

    }

}
```

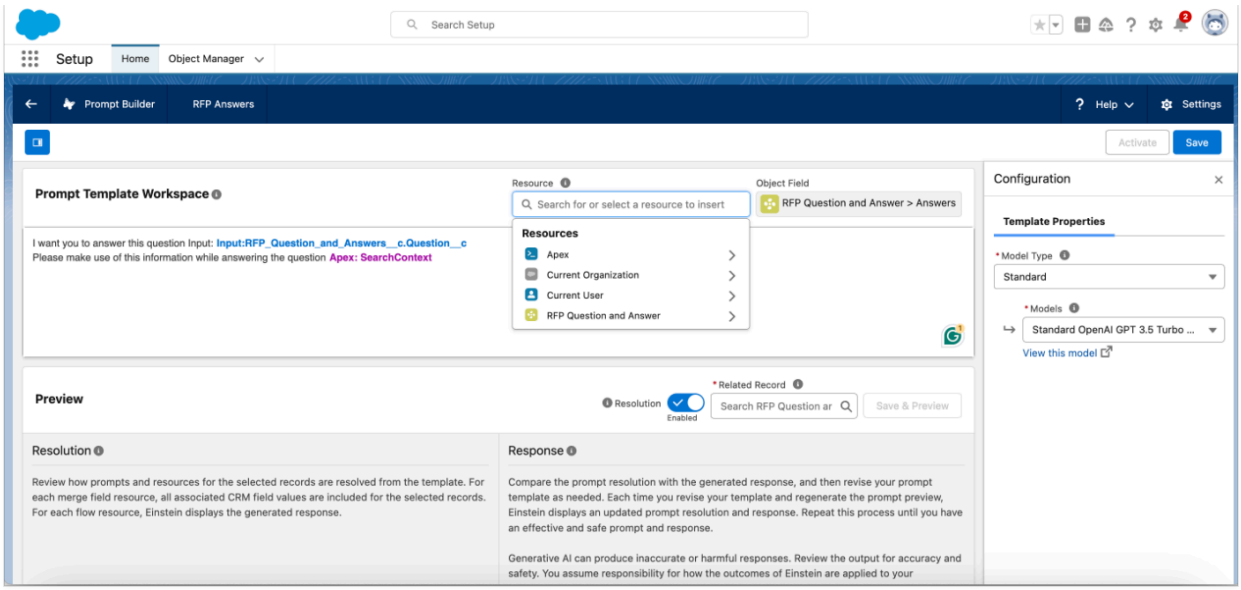
16. Navigate to setup → prompt builder → new prompt template



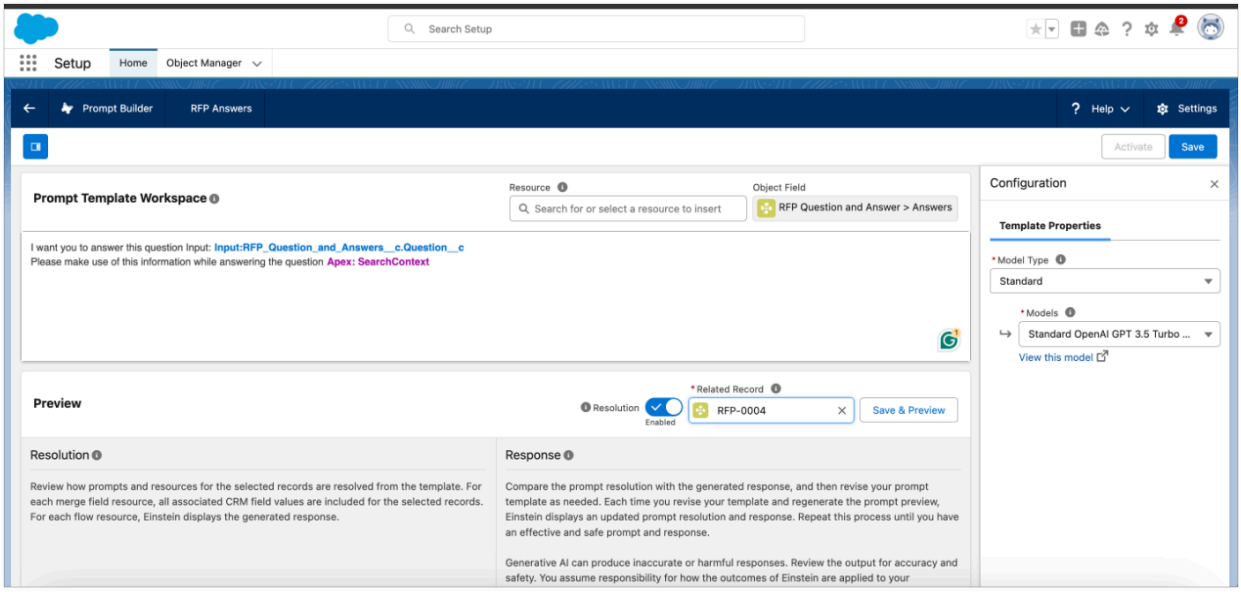
17. Enter the name, primary object, and the field for which you would like to generate content



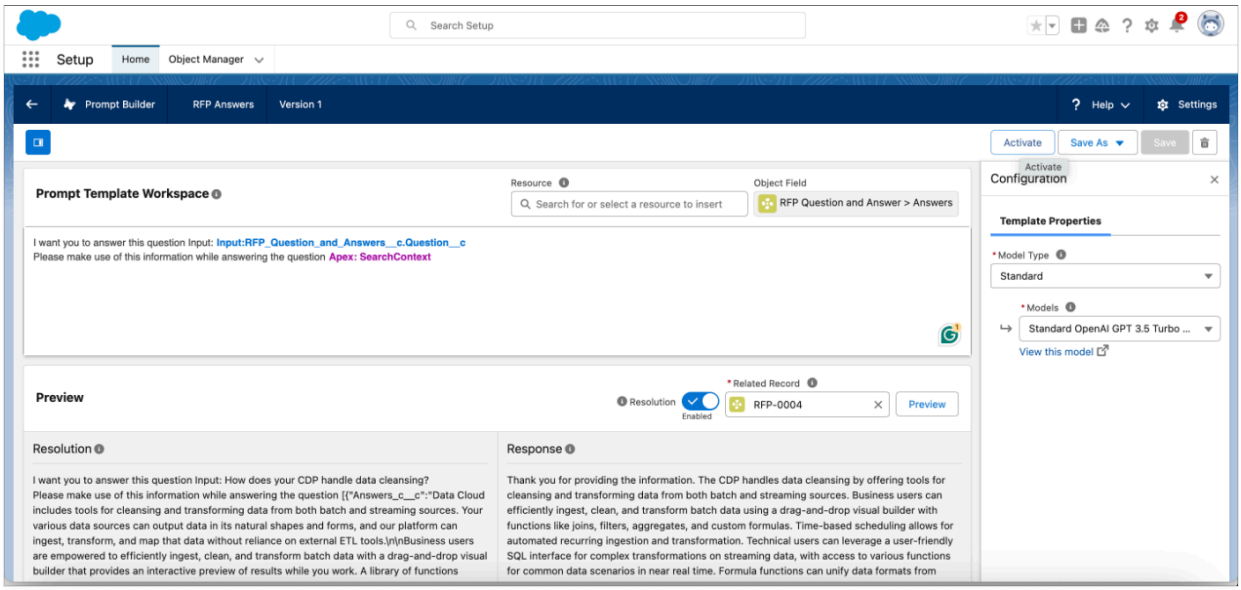
18. Draft prompt and specify resource



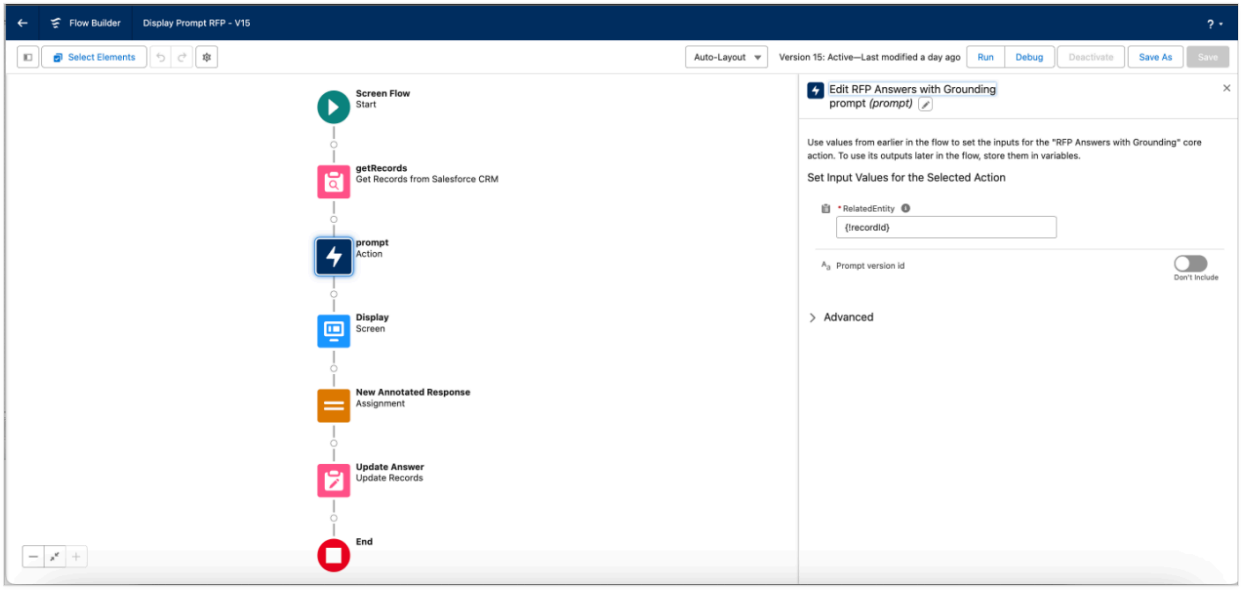
19. Save template, test prompt by selecting sample related record.



20. Activate template



21. Create flow to surface Gen AI responses



21. Create an apex class to find similar question and answers

```
global with sharing class SearchSimilarQandA {

    @InvocableMethod(label='Search Similar Questions' callout=true description='Perfor
    public static List<Results> query(List<Requests> requests) {
        List<Results> results = new List<Results>();
        Results result = new Results();

        String topic = requests[0].topicOne;
        System.debug('Searching Vector Database for topic: ' + topic);

        ConnectApi.CdpQueryInput input = new ConnectApi.CdpQueryInput();
```

```
        input.sql = 'select v.score__c Score__c, r.Id__c Id__c, r.Question_c__c Questi

        ConnectApi.CdpQueryOutput output = ConnectApi.CdpQuery.queryANSISql(input);
        List<RFP_Question_and_Answers__c> questionAnswers = new List<RFP_Question_and_

        for (Object searchRecord : output.data) {
            Map<String, Object> myMap = (Map<String, Object>) JSON.deserializeUntyped(

            // check for access of case record for the current user
            if (getUserRecordAccess((String) myMap.get('Id__c'))) {
                RFP_Question_and_Answers__c qa = new RFP_Question_and_Answers__c();
                qa.Question__c = String.valueOf(myMap.get('Question_c'));
                qa.Answers__c = String.valueOf(myMap.get('Answers_c'));
                questionAnswers.add(qa);
            }
        }

        result.searchResults = questionAnswers;
        results.add(result);
        return results;
    }

    public static boolean getUserRecordAccess(String recordId) {
        // Implement your logic for checking user access
        return true;
    }

    global class Requests {
        @InvocableVariable(label='Topic' description='Topic to search for' required=tr
        global String topicOne;
    }

    global class Results {
        @InvocableVariable(label='Search results' description='Search results of your
        global List<RFP_Question_and_Answers__c> searchResults;
    }
}
```

21. Create flow to surface the similar question and answers

RFP Demo with Vector Databases and Prompt Builder copie