# {'sentence\_0': {'text': 'Hello, guys. We are going to continue the lang chain series. And in this specific videos and in the series of upcoming videos, we are going to discuss about RAG pipeline. Now RAG, the full form is retrieval augmented generation. One of the very important use cases you may probably solving when you specifically use LLM models, and most of the use cases right now in companies are demanding this kind of skill sets, you know, where you can actually develop rank.', 'start': 0.56, 'end': 28.065}, 'sentence\_1': {'text': "Let's say you have a set of documents and you should be able to query that specific documents and get the result quickly. If you have different kind of files, let it be a read me file, a text file, a different data source file, you should be able to read it, convert that into a vectors, and able to retrieve any queries specifically from those kind of data sources. So, as I said, we will be implementing this completely from scratch, from basic to advanced. In this specific videos, we'll understand the entire architecture, and then we will do a lot of practical intuitions. Okay?", 'start': 28.445, 'end': 60.505}, 'sentence\_2': {'text': 'So, in any rack pipeline, these are the major components that will probably exist. So the first component is specifically called as load data source. Now whenever we say load data source, initially, we may have see, RAG is all about querying from a different different data source altogether. So we may be having files that may be having PDF. It can be a MD file, readme file.', 'start': 61.065002, 'end': 85.954994}, 'sentence\_3': {'text': 'It can be a Excel file. It can be a TXT file. It can be a database file. It can be different, different files altogether. Right?', 'start': 85.954994, 'end': 93.08}, 'sentence\_4': {'text': 'And in the first, so first step that we see over here, it is basically called as load source data. This step is also called as data ingestion. Okay. Data ingestion. Now in lang chain, the most amazing thing is that it definitely have a lot of different kind of data ingestion tools which will be able to load the data in various manner.', 'start': 93.46, 'end': 115.295}, 'sentence\_5': {'text': 'So in our current video, we are going to implement each and every of this component from data ingestion till the query vector store. Okay. Then after ingesting the data, you can probably do load. You can transform and you can embed. Okay.', 'start': 115.295, 'end': 130.02}, 'sentence\_6': {'text': 'We will discuss about this. What exactly is load, transform, and embed? Specifically, when we do loading, that is nothing but we are reading from a specific data source. Then we perform some type of feature engineering over here. If we want, like in transform stage here, the data, the complete data will be broken into smaller chunks.', 'start': 130.02, 'end': 150.725}, 'sentence\_7': {'text': 'Okay. Now why do we divide this data into smaller chunks? It is very much important to understand because whatever LLM models we specifically use, right, is definitely has some kind of context size. Right? So based on this context size, this is always a good good way to basically convert this entire data, which can be of many number of PDFs.', 'start': 151.205, 'end': 177.15999}, 'sentence\_8': {'text': "It can be, many number of pages in this specific PDF. We will try to divide this particular thing into chunks of data. So this is also what we'll going to see in the practical way. Then finally, we go ahead with embeddings. Embeddings basically means how we can convert all these chunks into vectors.", 'start': 177.15999, 'end': 195.135}, 'sentence\_9': {'text': 'Okay. How we can actually use vectors in order to probably convert this chunks, right? And finally, all these vectors will be further stored in some kind of vector stored database. Okay. So it can be a database over here.', 'start': 195.515, 'end': 210.885}, 'sentence\_10': {'text': 'And the main reason of this specific database is that we will be able to query this particular database in an efficient manner, right? With respect to any query that we have. Right? So whatever vectors are basically stored, if I have a query, if I hit the query on this particular database, I should be able to get the result based on the context of the query. So this is the entire rack pipeline that we specifically use.', 'start': 210.885, 'end': 235.99}, 'sentence\_11': {'text': "In the upcoming series of videos, we are going to use different different vector databases in different different clouds also. Okay? So right now we will try to implement all these things that you can actually see in front of you, and we will follow this entire architecture. So let me quickly open my Versus code. And once I probably start my Versus code, I will be, I will continue with respect to the same, thing that I've actually used.", 'start': 236.345, 'end': 261.42}, 'sentence\_12': {'text': "Right? All the projects that I've actually created. As I said that all the things will be given in the description of this particular video with respect to the link. Okay? Now I've created a folder which is called as rag.", 'start': 261.42, 'end': 272.355}, 'sentence\_13': {'text': "Inside this, I will create 1 IPYNB file. Okay? Initially, we'll go with IPYNB file, and then later on, we'll try to create an end to end project. Okay? So here, I will say, simple rag dotipynb.", 'start': 272.355, 'end': 285.485}, 'sentence\_14': {'text': 'Since we are doing it completely from basics, it is always good that we try to do it completely from basic itself. Right? So, this is the thing. So first of all, I will go ahead and see whether in my terminal, whether I have that Iperic kernel installed or not. Okay?', 'start': 287.145, 'end': 303.275}, 'sentence\_15': {'text': "So I will go ahead and write pip install IPy kernel. IPy kernel is specifically used to install the Jupyter notebook kernels itself. Right? So once I probably install this, you'll be able to see that this installation will take place, and I think it is already satisfied. So it is good enough.", 'start': 303.675, 'end': 321.115}, 'sentence\_16': {'text': 'We can probably go ahead and start the coding over here. Then I will go ahead and select my kernel, and it is nothing but 3.1, 10.0. Right? This is what is the environment that I have created. Now if I go ahead and execute something, so it is giving me some kind of error.', 'start': 321.115, 'end': 336.40002}, 'sentence\_17': {'text': "Okay? So perfect. Now what I'm actually going to do, I can see that there is some kind of message over here. I will just try to, disable this. I've already installed, this extension so that it'll not give us any disturbance.", 'start': 336.46002, 'end': 349.67}, 'sentence\_18': {'text': 'So let me do one thing. Let me just open over here for you and here I will open this particular folder and then I will go ahead and close this. Okay. Close window. Okay.', 'start': 350.13, 'end': 362.77002}, 'sentence\_19': {'text': "And now let me quickly open the Versus code again and here is my entire project. We will just try to execute it once again. Okay? Perfect. So that so that I don't have any disturbance with respect to that.", 'start': 362.83002, 'end': 377.475}, 'sentence\_20': {'text': "Okay? Now perfect. Now this is done. Now let's go ahead and start our coding. Initially, I will try to show you multiple data ingestion steps.", 'start': 377.475, 'end': 386.73}, 'sentence\_21': {'text': "Okay? So data ingestion basically means, let's say I want to read from a text file. I want to read from a PDF file. I wanna read from a web page. How can I actually do it?", 'start': 386.73, 'end': 396.78}, 'sentence\_22': {'text': "Okay. So all those things we'll start, to discuss. So first of all, I will go ahead and import from land chain underscore community dot document loaders. Now document loaders has all the techniques, specifically if you want to load, let's say a PDF, you want to load a Excel file, you want to load a TXT file. So over here, all the libraries will be present.", 'start': 396.78, 'end': 418.72}, 'sentence\_23': {'text': "So for umlangchin\_community.documentloaders, until then I will just close this terminal because I will not require over here. Then the first one that I'm actually going to discuss is about text loader itself. Now over here what I'm actually going to do, I'm going to create a loader is equal to text loader. And after creating this text loader here, I'm going to give a text file. Okay?", 'start': 418.935, 'end': 442.735}, 'sentence\_24': {'text': "Speech dot txt. So we'll just go ahead and see whether we have the speech dot txt file or not. No. I don't have it. So let me go ahead and create one speech dot txt file.", 'start': 442.79498, 'end': 453.415}, 'sentence\_25': {'text': "Okay? And, with respect to this particular file, what I am actually going to do, I'm to put some content. So I already have that content. Let me just quickly copy and paste it. This is one of the most famous speech, that is available in some history.", 'start': 453.555, 'end': 469.245}, 'sentence\_26': {'text': "It hasn't taken over here. So I just Googled it and I bought this particular speech over here and saved in speech. Txt. Okay. Then what I will do, I will go over here and read the speech.", 'start': 469.245, 'end': 480.32}, 'sentence\_27': {'text': "Txt. Okay? Now just by using this text loader, what will happen is that you'll be able to read the speech. Txt. Then I will use loader dot load.", 'start': 480.32, 'end': 491.49}, 'sentence\_28': {'text': "And here, I'm going to specifically convert this entire thing into text documents. Okay? Once I write loader dot load, that basically means it is going to just convert this into a text documents. And this finally you'll be able to see my test documents, and I will go ahead and execute it. So here you'll be able to see that once it is reading the specific speech, it has the entire text document.", 'start': 492.03, 'end': 519.685}, 'sentence\_29': {'text': "Now it is becoming so easy just to read the TXT file. Okay? Now the next thing that I'm going to specifically do is that, since I'm also going to use open API keys or, I'm also going to use Olama Embeddings or Ollama model. So what I can do is that I will just go ahead and import OS. The next thing that I will go ahead and do is law load underscore dotenv.", 'start': 519.685, 'end': 543.43005}, 'sentence\_30': {'text': "Okay? So load underscore dotenv. So for this, I'm also going to import from dotenv. Import load\_.env. Okay?", 'start': 544.21, 'end': 556.955}, 'sentence\_31': {'text': "Now why I'm doing this so that I will be able to call all environment variables and in one of the environment variables, I also have my open API key. So now I will go ahead and write os.environment. And here I'm going to use my open\_a, sorry, openai apikey. And here I'm just going to use os.getenv. And here again, I'm going to use my open API key.", 'start': 557.09503, 'end': 590.15497}, 'sentence\_32': {'text': 'So this is basically going to call the open API key from the environment variable. Now a altogether over here, guys. Now is the main term that we will be starting to, see some more data ingestion techniques. So one more data ingestion technique is directly reading from the web based web based, loader. You can basically see.', 'start': 590.15497, 'end': 613.22}, 'sentence\_33': {'text': 'Okay. And for this, again, I will be importing. Understand any document loader will be available in this specific library itself. And instead of writing text loader, we will be using web based loader. Okay?', 'start': 613.22, 'end': 626.165}, 'sentence\_34': {'text': 'Now along with this, I will also import bs 4. That is Beautiful Soup 4. So for that also what I will do, I will just import over here in my requirement dot txt bs4 so that whenever I execute this, it should be able to run. Okay? Now in order to do something like this, load chunk and index the content of the page, HTML page, right, or any web page.', 'start': 626.545, 'end': 654.5}, 'sentence\_35': {'text': "So here I will create a loader and this will be equal to web based loader. And here I'm going to give 2 important parameters. 1 is the web path. So web\_paths. Okay.", 'start': 654.5, 'end': 668.32996}, 'sentence\_36': {'text': 'This is culture. And here, I will give the URL. So let me just take one URL from the GitHub IO page that is also available in the documentation of Langchain. So this is what is the page over here. What I will do is that quickly I will open a browser and show it to you how this page looks like.', 'start': 668.32996, 'end': 686.45996}, 'sentence\_37': {'text': "Okay? So if I execute this over here you'll be able to see that. Just a second. Okay. Perfect.", 'start': 686.45996, 'end': 695.025}, 'sentence\_38': {'text': 'So here you, you are able to see that. Just a second. The seed is this the same page? Yes, it is the same page. And I will be opening this.', 'start': 695.23, 'end': 705.335}, 'sentence\_39': {'text': 'Okay. Perfect. So this is the page that you will be able to see over here. And I want to probably read this entire page and, use it as a rag system itself. Right?', 'start': 705.395, 'end': 716.43}, 'sentence\_40': {'text': "So in order to read this entire page, I can just take this URL and use this web loader library. So here, if you see, I'm using this web based loader library and I'm giving the 1st parameter that is the web path. Okay. The second parameter that we can specifically give is our arguments. Right?", 'start': 716.49, 'end': 735.195}, 'sentence\_41': {'text': "So here I will rate bs.\_kwargs. Okay. So this is the second parameter. I'm just seeing the documentation and we will give the next argument in the form of dictionaries. The first is parse underscore only.", 'start': 735.515, 'end': 749.65}, 'sentence\_42': {'text': "Okay. And then here I'm basically going to use my BS 4. That is a beautiful soup for soup trainer. Okay? Soup strainer, soup strainer.", 'start': 750.19, 'end': 762.99}, 'sentence\_43': {'text': 'Okay. Yeah. Soup strainer. So this is the first, parameter that I will be specifically giving And understand why this is required because here, the soup strainer and since we are using beautiful soup, over here, we really need to give the classes that it re it needs to read from that particular page. Okay?', 'start': 763.29004, 'end': 783.72003}, 'sentence\_44': {'text': "And here, what all classes are there? Let's go ahead and see this. Okay. So if I probably see over here and if I do just inspect element, okay, so here you can see post title is there. Then you have something like post content where all the content is basically available.", 'start': 784.02, 'end': 801.4}, 'sentence\_45': {'text': "So let us take this one as my first one. Okay. So I will just go ahead and execute this. And here you'll be able to see that, this will basically be my post title. And here you have the post underscore content.", 'start': 801.46, 'end': 817.13}, 'sentence\_46': {'text': "And third one that I also want to take is post underscore header because header will also have some information over here. Right? So this is done. You'll be able to see that it it looks absolutely fine. And here, with respect to this, we have also created this entire loader.", 'start': 817.67, 'end': 833.0}, 'sentence\_47': {'text': "And here, I will use a comma so so that this is basically my argument over here. Right? So once we do this, then let's see whether it'll execute or not. BS 4 is not found. As I said that BS 4 is not there.", 'start': 833.46, 'end': 848.33997}, 'sentence\_48': {'text': 'So let me do one thing quickly. Let me open my command prompt and do the pip install requirement dot txt. Okay? So here I will go ahead and write pip install -rrequirement.txt and execute it over here. You will be able to see now BS 4 will also get installed.', 'start': 848.48, 'end': 865.77}, 'sentence\_49': {'text': 'And once it is getting installed, it is completed. It is good to go. Now if you go ahead and execute it, I think it should work. Perfect. Now loader is there.', 'start': 865.99005, 'end': 874.40497}, 'sentence\_50': {'text': "Now what we are going to do over here is that again, write loader dot load. And finally, I'll be able to get in the form of text documents, text\_ documents. Okay? Documents. And here is my text document.", 'start': 874.40497, 'end': 889.195}, 'sentence\_51': {'text': "So perhaps you meant, this one. Let's see. So guys, there was one one error. We need to give this URL in the form of tuples. So let this get closed and then I will write comma over here.", 'start': 889.195, 'end': 903.755}, 'sentence\_52': {'text': "Now I think it should be working. Let's see. Yes. Perfectly. It is working right now.", 'start': 903.755, 'end': 908.575}, 'sentence\_53': {'text': 'And if you go ahead and see your text documents, again, you will be able to get all the information from that particular web page in the form of URL. Okay? Or in the form of documents. So this is perfect. You are able to get every information over here.', 'start': 908.635, 'end': 922.53503}, 'sentence\_54': {'text': "And, with respect to this, you will be able to see that, yeah, you have all the information and you can also see the page content and everything looks fine. So these are some of the ways how you can specifically make sure that, you probably get all the content from a page itself. One more thing, this is not underscore, this is dash. So now I think you'll be able to see more content. Okay?", 'start': 922.53503, 'end': 944.945}, 'sentence\_55': {'text': "So let me just open this. So here you'll be able to see all the content itself. So, this is one more way. One more way is directly to read from the PDF itself. So what I will do is that I will quickly create a PDF.", 'start': 945.245, 'end': 958.02496}, 'sentence\_56': {'text': "Okay. So let me do one thing and I will upload a PDF over here. Okay. So here I've directly read completely from the PDF itself. Okay.", 'start': 958.08496, 'end': 972.1614}, 'sentence\_57': {'text': "And how to do that? See, still we are in the data ingestion phase. And how do we do this? We'll also see that. So from, the same library that we'll be using, again, we have to focus on document loader.", 'start': 972.1614, 'end': 984.36}, 'sentence\_58': {'text': 'And here I will copy and paste it. Okay. And this is basically my PDF reader. One more library dependency will be there. Okay?', 'start': 984.74, 'end': 995.095}, 'sentence\_59': {'text': "So we will try to install that also. So here I'm going to write pypdfloader. Okay? And, again, I will be using loader is equal to pypdfloader. Py pdfloader.", 'start': 995.095, 'end': 1007.925}, 'sentence\_60': {'text': 'And I have to give my PDF name. Right? So PDF name over here specifically is nothing but attention dot pdf. So it is in the same same folder itself. So here I will write attention.', 'start': 1008.225, 'end': 1019.38995}, 'sentence\_61': {'text': "Pdf. Okay? Now I think I should be getting an error. If I don't get an error, I will definitely install. Yeah.", 'start': 1020.25, 'end': 1026.1849}, 'sentence\_62': {'text': "I'll install it now. See, pip install pypdf. So here I will make sure that I will write requirement. Txt, and I will write pypdf. Okay?", 'start': 1026.1849, 'end': 1035.19}, 'sentence\_63': {'text': "So all these things you really need to take care of because there will definitely be dependencies. Because and understand, RAG is one very important thing that really needs to get created in the form of application. So that is the reason I'm creating this completely from scratch. So pip install minus r requirement dot txt. So this is also done, I think.", 'start': 1035.19, 'end': 1053.39}, 'sentence\_64': {'text': 'And pypdf is also installed. Perfect. Then the next step, again, I will go ahead and execute this. And but here also the same step will be required, loader dot load. And this will basically be my documents.', 'start': 1053.39, 'end': 1064.995}, 'sentence\_65': {'text': "And I will execute this. Once I probably execute this, this is my docs. Okay? Perfect. So here you'll be able to see my entire PDF has been read and it is in front of you.", 'start': 1065.535, 'end': 1075.24}, 'sentence\_66': {'text': 'Now the next step. Now see, this load data source part is basically done. We know how to load it. Okay? Now you will move towards this transform.', 'start': 1075.335, 'end': 1083.035}, 'sentence\_67': {'text': "Okay? Then we will go ahead towards embed. Okay. So the first part of load I've actually done with respect to PDF web based with respect to text file. If you go ahead and check out the Langchain documentation, this there are still more amazing ways to do for Excel file, for read me file, different directories even for directors.", 'start': 1083.495, 'end': 1102.755}, 'sentence\_68': {'text': "Let's say that you have a lot of files in the directory itself, many files, and you can also load that. So we'll try to see in upcoming videos more about different different examples. But here, just to give an idea about load data source and load is actually done. Now let's go ahead and do this transforming. Now transform is very much important.", 'start': 1102.755, 'end': 1120.64}, 'sentence\_69': {'text': 'Now understand, guys, this is your entire PDF documents. Right? Now you need to convert this PDF documents into chunks. Now how do you do that? And again, there are multiple ways.', 'start': 1120.86, 'end': 1130.715}, 'sentence\_70': {'text': "And this way is entirely falls inside the category of text splitter inside langchain. So here we are to write from langchain from langchain.textsplitter import recursive recursive character text splitter. We'll try to split it with the with the help of text itself. So here I'm going to create my text splitter as a call to a recursive character text splitter. And here I'm going to specifically use my chunk underscore size is called 2,000.", 'start': 1131.415, 'end': 1167.425}, 'sentence\_71': {'text': "I want this specific size to be the chunk. And let's say the overlap chunk chunk overlap. Okay? Chunk overlap. So here I will go ahead and write chunk underscore overlap.", 'start': 1168.365, 'end': 1183.395}, 'sentence\_72': {'text': "This is called to 200. I'll keep this as 200. Okay? So once I do this, now this text splitter will be, responsible text\_splitter.split documents. We'll be responsible in splitting all the specific documents that I give over here.", 'start': 1183.615, 'end': 1199.5249}, 'sentence\_73': {'text': "Okay? So this docs will be there and this will be my documents, final documents that I will be able to see. Okay? Let's display this documents, the top 5 documents still here. Okay.", 'start': 1199.5249, 'end': 1211.4501}, 'sentence\_74': {'text': "And let's see this. So here you'll be able to see provided proper attrition is Google by grant permission, best model, this, this, this, this, everything is there. Attention not PDF. Right? So all the information is basically there.", 'start': 1213.875, 'end': 1225.9}, 'sentence\_75': {'text': 'And with respect to this, you really want to see the more things, more different, different, all the documents in short. So you can also go ahead and see the entire documents over here. Okay. Done. So these are all my documents over here.', 'start': 1227.08, 'end': 1241.0701}, 'sentence\_76': {'text': 'Transformers, everything is available over here. Now see, this is my entire documents. It has been divided into proper smaller chunks. Now we can take the chunks, and we can convert that into vectors. Okay?', 'start': 1241.0701, 'end': 1252.7151}, 'sentence\_77': {'text': "And that is what has given you an idea over here. Right? Like, how we have transformed. We have basically taken this entire PDF document and we have divided that into chunks. Now it's a time that we understand how to probably convert this into vectors.", 'start': 1252.775, 'end': 1267.245}, 'sentence\_78': {'text': 'And for this, we will also be using some different different vector, vector embedding techniques. Right? One of the embedding techniques that I will be showing you is with respect to OpenAI. So here, now we will go ahead and probably write about vector embeddings and vector store. Right?', 'start': 1267.545, 'end': 1285.665}, 'sentence\_79': {'text': "Vector embeddings is a technique wherein we convert a text into vectors. Okay? So how do we do that? Again, with the help of OpenAI. So here I'm going to write from Lang Chen Endoscope Community dot embeddings, import OpenAI OpenAI Embeddings.", 'start': 1286.685, 'end': 1304.565}, 'sentence\_80': {'text': "You can also use OLAMAI Embeddings. It is up to you since if you don't have OpenAI API, you can use Ollama Embeddings directly. But the performance of OpenAI Embeddings is better, far more better than Ollama Embeddings. Okay? So here, what I'm actually going to do next thing is that, first of all, we need to understand how to probably create vectors.", 'start': 1304.565, 'end': 1323.7899}, 'sentence\_81': {'text': 'Again, to create vectors, we can use open AI embeddings. But after creating the vectors, we also need to store in the vector store. That is what it is. Right? This vector store is like a kind of a database.', 'start': 1324.01, 'end': 1335.365}, 'sentence\_82': {'text': 'Embeddings can be embedding is a technique where you convert text into vectors, but later those text needs to be stored in some kind of vector store. Right? So for this reason, we will be using something called as Chroma DB. So they are come, there are couple of, there are couple of vector databases that has been provided by Lankan itself. One is Chroma.', 'start': 1335.745, 'end': 1355.765}, 'sentence\_83': {'text': "1 is Fias. As we go ahead, you know, how to create this kind of vector database in the cloud also, I will show it to you. So from langchain\_community, I will be using dot vector stores. And here I'm going to import 1 is nothing but chroma. So with respect to this, we will go ahead and create my DB.", 'start': 1355.765, 'end': 1374.32}, 'sentence\_84': {'text': "And here I'm going to write chroma.from\_documents. Okay. From documents. And here, I'm going to basically give my entire document. Okay?", 'start': 1374.46, 'end': 1387.395}, 'sentence\_85': {'text': "Entire documents. And I'll not give the entire document. Let's just give the first 20 documents itself because it will take more time to create the embeddings. Right? And here, the embeddings that I'm going to specifically use is nothing but OpenAI embeddings.", 'start': 1387.535, 'end': 1401.1549}, 'sentence\_86': {'text': "Once I execute this, I think I'm going to get an error saying that Chroma is not available. I don't know whether I've installed Chroma or not. So here you can see pip installed Chroma DB. I rarely require Chroma DB. So again, I'll go back to my requirement dot txt.", 'start': 1401.375, 'end': 1413.66}, 'sentence\_87': {'text': "The reason why I'm showing you all this completely from scratch so that you whatever error you face, you should be able to fix it up. Okay? So I'm going to delete this, and let's go ahead and write pip install. One more library that I'll be using is nothing but Fire's CPU. Okay?", 'start': 1413.88, 'end': 1431.53}, 'sentence\_88': {'text': "Fire's CPU because Fire's is also one type of chroma one type of vector database. Okay? So minus r requirement dot TXT. So let's install both these specific libraries. It'll take some amount of time.", 'start': 1431.91, 'end': 1443.5801}, 'sentence\_89': {'text': "Again, it depends on your internet speed and how fast your system is. But, yes, I think, Fyce CPU and Chroma DB is the kind of, vector databases that we'll be using. One assignment I'll give you, try to also use LAN's vector database from, again, seeing the Lang Churn documentation, you can actually do it. Okay? Now this is done.", 'start': 1443.5801, 'end': 1463.56}, 'sentence\_90': {'text': "Once, this will probably be done, then we can go ahead and check by executing it whether it is working now or not. But at the end of the day, here, what we are actually doing, we have imported OpenAI embeddings. The vector database that we are going to use is Chroma. Then chroma dot from document, I'm giving the entire document and using these embeddings, it'll be storing inside this particular vector store. Now this DB, we can store it in our local.", 'start': 1463.56, 'end': 1488.04}, 'sentence\_91': {'text': "We can store it in the cloud wherever you specifically want. Okay? So, guys, the installation has been completed. Now let's go ahead and execute this Chroma part. Now here you'll be able to see that entire embedding will basically happen.", 'start': 1488.04, 'end': 1500.04}, 'sentence\_92': {'text': 'And again, we are going to use the open AI embedding for this. And now this DB is nothing but our vector database. Okay? So if I probably consider vector database, and now all I have to do is that I can query anything with respect to this particular vector database to retrieve any kind of result that I really want. So I will create a query, first of all.', 'start': 1500.04, 'end': 1519.6499}, 'sentence\_93': {'text': "So here, let me go ahead and write my query. So query is like I will go ahead and write, who are the authors? Who are the authors of attention is all you need, research paper. Okay? So this is my query that I'm going to specifically ask.", 'start': 1519.6499, 'end': 1537.85}, 'sentence\_94': {'text': "Let's see whether it will be able to understand this or, and it'll be able to give us the result. So here I will go ahead and write db. Similarity search. Now there are multiple options. Similarity search, similarity search by vector if you want.", 'start': 1538.0549, 'end': 1552.36}, 'sentence\_95': {'text': 'You can also convert your data into by using, again, the OpenAI embeddings, and you can query it from here. But we are just going to use the similarity search since we are just going to use the query over here. Right? So if I write dv. Similaritysearchofquery, so this will basically be my result.', 'start': 1552.42, 'end': 1567.9901}, 'sentence\_96': {'text': "Okay? Now let's go ahead and see my result. Okay? And let's execute this. So here you'll be able to see that I'm able to see multiple information, like 4 documents has been over here.", 'start': 1568.3701, 'end': 1579.95}, 'sentence\_97': {'text': "So let's take the first document, and let's see how the result is. So here I'm going to basically take 0. And inside that, the field name is nothing but page content. So I'm going to basically write my page content. Now here you'll be able to see prop provided proper attribution is provided Google here by grant permission.", 'start': 1579.95, 'end': 1597.31}, 'sentence\_98': {'text': 'Google brain, all the email IDs, all the researchers names are actually available. So that basically means it is able to research, it is able to retrieve the results of the all the scientists who are involved, all the researchers who are involved in creating this particular paper. I can also do one more thing. I can go ahead and write what is attention, is all you need. Okay?', 'start': 1599.7101, 'end': 1622.98}, 'sentence\_99': {'text': "Attention is all you need. Okay? So let's see what kind of results I will be able to get. And understand this is entirely rack pipeline, which is being able to come from the entire documentation. And here you'll be able to see that result is also coming.", 'start': 1623.68, 'end': 1638.14}, 'sentence\_100': {'text': "Let me do, write something which is available in the research paper. So let me open my research paper over here, and let me ask some question directly. Okay? From here, you will be able to see. See, when I'm searching attention is all you need.", 'start': 1638.84, 'end': 1652.76}, 'sentence\_101': {'text': 'It is coming from here itself. Right? So let me use something over here. Let me just try it over here. Encoder is, what is an attention function?', 'start': 1652.76, 'end': 1666.585}, 'sentence\_102': {'text': "Let me just go ahead and search this. Okay? Let's see. Attention is something, something, some text is there over here. So if I go ahead and execute it, and here you'll be instead of performing a single attention function with a model dimension, values, keys, and all the information is basically coming up.", 'start': 1666.805, 'end': 1682.9149}, 'sentence\_103': {'text': 'Right? And here is a very good result. And this is directly coming from the vector database. That is the most amazing thing over here, right? Now you may be also thinking, Krish, can we also use the Fyze database, the Fyze directory database that we have used, right?', 'start': 1682.9149, 'end': 1695.44}, 'sentence\_104': {'text': "So let me just show you with respect to that also. Fyze. And here I'm going to use the Fyze vector database. And this will also give you an idea how you can actually store that embeddings into Fios database itself. And we'll also do that.", 'start': 1695.44, 'end': 1709.16}, 'sentence\_105': {'text': "So here I'm going to write from langchain langchain\_ community.vectorstores. I'm gonna import my FAIS files, and then I'm going to also use tb is equal to fais.fromdocuments, the same thing. Right? So from\_ documents. And here, I'm gonna use my documents.", 'start': 1709.16, 'end': 1737.365}, 'sentence\_106': {'text': "Okay? And let's say I go ahead and do the embedding for the first 20 documents, and then here also I'm going to use my OpenAI embeddings. And I think this is also my another DB, so I will write it at DB 1, which is my files database. Okay? From documents.", 'start': 1737.985, 'end': 1754.365}, 'sentence\_107': {'text': 'Okay. No worries. So here the spelling mistake was there and I think this will also work. Now DB one is also ready and what I can do, I can use the same thing and paste it over here. Okay.', 'start': 1754.425, 'end': 1765.06}, 'sentence\_108': {'text': "And just try d dvone.similaritysearch. And here you can see instead of performing a single attention, everything is probably coming up. So I've just shown you the example of both Fias and, Chroma database, Chroma vector database. As I said, one assignment will be given to you as regarding Lance vector database. You can go and search it for that in the in the in the Lan Chien documentation.", 'start': 1765.2001, 'end': 1789.7999}, 'sentence\_109': {'text': 'But here, if I probably see with respect to this, we have completed this load, transform, embed, then we have created this vector store. And then we have queried that particular vector store and retrieved the most similar results. Right? So this is just a beginning of creating a rack pipeline. Now after this, in the next video, we are also going to discuss about retrievers, chains, how we can convert this, how based on context we can retrieve different different data.', 'start': 1790.34, 'end': 1813.485}, 'sentence\_110': {'text': "All those things we'll be specifically discussing about. So I hope you like this particular video. This was it from my side. I'll see you on the next video. Have a great day.", 'start': 1813.485, 'end': 1819.405}, 'sentence\_111': {'text': 'Thank you all. Take care. Bye bye.', 'start': 1819.405, 'end': 1820.9437}}