handling database schema changes is an important part of web development and in this video we're going to look at an excellent goal package for working with database migrations and that package is called Goose now database migrations allow you to manage how the database schema changes over time and a tool like goose is going to allow you to add new changes to that schema and it's also going to allow you to roll back to a previous version of the database so I'm on the documentation now Gus is a database migration tool as it says here let's cross over to the GitHub page and I'll leave a link to these below the video Gus supports SQL migrations and also go functions we're going to see a little bit more about the SQL migrations very soon in this video now before we dive in if you want to support the channel we have this coffee page linked in the description of the video thanks very much to everyone that's contributed to that so far and on another note if you want to see more go content let me know in the comments what you would be interested in seeing some of the most popular videos on the channel are Go videos but we've only done a couple of them most of the videos of course are Python and Django related if you want to see more go just let me know in the comments what you'd be interested in now let's get started by opening vs code and we have a very simple main. go file here that just prints High to the terminal when it's run now what we're going to do to start with is install Goose now you can get the installation by going back to GitHub and if we scroll down here to the readme section there's an installation section here let's start actually with a look at the top here so we can manage the database schema by creating incremental SQL changes or go functions and Gus is going to work with multiple data bases for example postgres MySQL SQL light and many more now if we want to install this we can go to this installation command and it uses the go install command so we can copy that and this is going to install the ghost binary into your gopath /bin directory and if that binary is too big and you want to have a smaller binary you can build a light version by excluding any drivers you don't need so for example if you're not going to use postgress you can pass this tag of no postgress and the same applies to other databases as well you can pass those tags when you run the go build command and that will help cut down the size of the binary so let's open vs code and I'm going to paste in the go install command here now I've actually already installed Gus into this project so if we run the Gus command you can see we get the different sub commands of the package now the important command if you want to actually migrate the database to the most recent version available is the goose up command and as well as the most recent version you can also use the up by one command and we're going to see examples of those in this video we're also going to look at the ghost down command that's going to roll back the database version by one and there's also a down to and up to option where you can specify a specific version that you want to migrate the database to now in this video we're going to use Docker and we're going to set up a postgrads database and the migrations that we Define here with Gus using SQL migrations we're going to apply those and that's going to change that underlying postgres database so that's going to be the workflow that we're going to go through in this video now if we go to the GitHub page we can go down to this usage section here and you can see those commonly used commands in order to actually create a new SQL migration we can use the Gus create command and when you use Gus create what you do is you give the migration a name for example you would call the file add some column and then you specify whether it's an SQL migration file or a go migration file now in this video we're going to stick with SQL migrations so let's get started and use the Gus create command in order to actually create a migration file now let's imagine that we have a database table for team and these could be sports teams for example or they could be teams in some kind of job or Corporation or something like that so let's clear the terminal and we're going to run the goose create command here I'm going to give this a name of add team table because what we actually want to do here is create a table in the database called team and I'm going to Define this as an SQL migration so if we look here it's created a new file in the current directory and you can actually specify the directory where you want these migration files to go that's an option we'll see later on but if we look at this file here we can see that there's some directives in the file at the top we have Gus up and then below that here we have Gus down and the SQL itself is surrounded by Gus statement begin and statement end and that applies to the bottom as well here as you can see now the up migration

that's going to define the change that you want to apply to the database with this migration so we want to create a table in the database called team so we can Define the SQL to do that and the g down command is going to roll that back so for example if we create the table here we want to roll that back if we run the ghost down command so let's see how to define the SQL for this migration file now notice that the name of the file has the name that I put into the command but it's prefixed by a Tim stamp and that time stamp is actually important because Gus can then read the migration files and it knows the correct order for those files and it can then apply them in the correct order to the database so let's get started with adding the SQL now I'm not going to write this out from scratch I'm just going to paste this in so what we have here is a create table SQL statement and we're creating a table in the database called team we're going to give that an ID and that's going to be the primary key and because we're using postes we'll use the serial data type and we're going to give the team table three other columns one for the name of the team one for the city that the team is based in and then finally we have a coach column so we're going to assume that this is a sports team and all three of these are varar fields and the first two of them have a not null constraint they cannot be null in the database so when we run Gus up we want to apply this migration and therefore create the table in the underlying database and when we roll This back what we want to do is drop the table so I'm going to go down to this statement here when we run goose down what we want to do is use the drop table statement in SQL and we're going to drop the team table and that basically reverses what we do here in the goose up command so let's save the file and before we can actually apply this migration what we need to do is set up the database so what I'm going to do is go to Docker Hub and we're going to to do this very quickly but I have a page open here for post Gres and if you're using Docker you can pull this down using the docker pull command and I've actually already done that as you can see here we ran Docker pool post GES and that image is now available on my computer now you don't need Docker for this video you can install post GES onto your machine and as long as you define the correct connection details then you can follow along with the video now if we scroll down here we get an example command on how to use the image that we've just pulled so we can start a post instance using Docker run and we can pass some details into that for example the password of the post user what I'm going to do is write an altered version of this command so let me go back to VSS code here and in fact I'm going to use the terminal for this so let me bring that back here what we're going to do is use the docker run command and let's give this a name of Gore DB and let's set the environment variable for the postgres password and we're going to set that equal to just secret and I'm also going to add a port mapping here that's going to allow us to connect to this post database from the host machine so we're mapping Port 5432 in the container to Port 5432 on the host and then we can pass DD and that's going to run this in the background and finally the name of the image was postgis and when we start that you can see that we have a container ID here and if we run the docker PS command you can see that postgres is now running so we have a postgres container that's available to connect to and to apply these migrations to that database now if we go back to the Gus get page here remember we had this Goose up command to apply the migrations but we will need to tell Goose where this database that we want to use is actually located now I'm going to go to a page on the Gus documentation for these commands and you can see in this usage section here that when you run a goose command you can specify the driver and the database string to actually use and that's going to tell Goose where the database is that you actually want to apply these migrations or roll them back so what we need to do is tell G that we're using a postgres driver and we also need to tell Gus where that database is using the database string so let's go back to VSS code here and what I'm going to do is run the Gus command and the name of the driver is going to be postgres and then we can pass a connection string to the postgresql database now I'm going to paste this connection string in here so we have postgresql and the user is postgres the password is secret and we're connecting a local host because we exposed Port 5432 inside the container to the host and we specify that port number here and the name of the database that we're connecting to is going to be postz so I hope that makes sense if you have used a different password in the docker run command you can specify that here and if you've set up a different user rather than the default postgres user you can also specify that as part of the database string now the final part of this command is what we actually want to run and that's the goose up command so we can

specify up at the end of this and then we can run this here and you can see we get some output below so we have the okay message here and this migration which which is the add team table migration that has been applied and Gus has now successfully migrated the database to that version now how can we actually check to see if we have that database what I'm going to do is use an extension for VSS code to look at the postr database on VSS code and it's this extension here you can see it's called post SQL and it provides a management tool that you can use on vs code now after you've added the extension you can go to the command pallet and that's with command shift p on Mac and you can use control shift p on Windows and if we type in this post SQL extension here we can add a connection to our database now the host name is just going to be Local Host and the user is going to be postgres and the password for that user remember was secret and we're using the standard postgres Port of 5432 and this is going to be a standard connection it's not going to be a secure connection and let's connect to that postgres database and the final thing we need to do is give it a display name and I'm going to call this ghost demo and once we've added that if we click the these icons here you can go to the postgresql Explorer and you can see the Gus demo database has been added here and we can connect to that now and we can do things like viewing the tables in that database and you can see there are two tables at the moment one is called ghost DB version and that's added by Goose itself and we also have a team table here that contains the four columns that we specified in the migration file so we have an ID a name a city and a coach column and you can see the data types as well these correspond to what we added in that SQL migration now let's say we've made a mistake here we can actually revert this migration so what I'm going to do is go back to the terminal and when we revert that we run the ghose down command and it will apply whatever SQL code that you've added in here so let's do this again and I'm going to bring back that last command which was the goose up command and all we need to do here is change the statement at the end so I'm going to change this so we can see this a little bit better and we're going to run the down command now when we run the down command you can see that we have okay message here and if we go to the left hand side what I'm going to do in this public schema in the postres database is refresh the items and you can see that the team table has now been removed from this postgres database so it's very easy to run the Gus up command if you want to apply new migrations and get it to the latest version but if you do need to go back in time and go to a previous version of your database schema you have the goose down command for that now I'm going to rerun the Gus up command and that's going to bring back that table at the left hand side site so again we can refresh the public schema and now we have the table so that's been readded to the database and we can now easily start building a history of SQL migrations in this go project by using the gust package now let's add a slightly more complicated migration so I'm going to run the G create command and if we look at the team table here let's imagine that this coach column that we have is now a new table itself so we don't want this to be represented by a string but we want a foreign key to the coach table so we're going to have a new table called coach and we're going to refactor this and change the database to actually use a foreign key in the team table so let's give this a name I'm going to call it add coach table with foreign key from team and again it's going to be an SQL migration so that creates a new migration file and if we go back to the file system here you can see that that file has been created here and again it's up to us as developers to Now define the Gus up and G down SQL statements so let's get started this is going to be a slightly more complex migration because the goose up command is going to have multiple statements we're going to start by creating that coach table with an ID and a name and then what we need to do is we need to change the team table and remember that has a column called coach which is just a varar field so one way to do this because I know we don't have any data at the moment is we can use the alter table command in SQL or rather the alter table statement and then we can drop the column coach from the team table and that might not be the way to do this if you already have data in the database but we just going to roll with it for this example now after we've dropped the coach column what we could do is again run an alter table statement on the team table in the database and we can add a new column called coach ID with the data type of integer and then finally we can add one more statement here one more alter table statement that is and we're adding a constraint called foreign key coach so it's going to be a foreign key for the coach ID column that we've just added above and that's going to reference the ID

column in that coach table that we created at the the beginning of this migration so that's the Gus up command it will create the table it will then drop the coach column from the team table in the database and add a coach ID column instead and then it'll link that coach ID column using a constraint and that's going to be a foreign key from coach ID in this table to the ID field in the coach table so now that we have the foreign key and we've got this Goose up command what we can do is Define the Gus down command as well now let's start by dropping that foreign key coach constraint again alter to team and then we use the drop constraint statement to drop that constraint and we basically need to reverse what we did above so we're going to add back a coach column and remove the coach ID column so let's start by removing the coach ID column using the drop column statement and then we can add back that coach column of type varar and of course the final thing to do is drop this table coach that we added at the start of the goose up statement so drop table coach and that's going to revert the database back to the previous state before this migration was run now let's go back to the terminal here and I'm going to clear this out and again let's run that goose up command that we had before so we run Gus here and we pass the post GES driver and then again we're passing that database connection string and we're running the up command at the end of that and that's going to upgrade the database to the latest version and if we go back to the postgres Explorer here on VSS code and if we look at what we have in the public schema if I refresh that you can see we now have a coach table with those two columns and if we go back to the team table here you can see we have the coach ID column and you can see that that's a foreign key to the ID column in the coach table so that's now worked but there is something I want to change here I'm going to clear the terminal I don't want to keep passing that database driver and database connection string into the command first of all it's very inconvenient but there is a serious problem with that as well we are exposing the password for this database user in the command line that's not something you typically want to do on your server you want this to come from some kind of environment variable and you don't want to have to actually write this password in plain text on the command line so how do we fix this well it turns out that Gus supports environment variables and there are some that we can use here for example Gus driver that's going to specify the database driver to use and importantly as well we have Goose database string that's going to have the database connection string and we can also specify a directory for our migration files and when we run the Gus create command to actually create a new migration file that particular file is going to be located in that migration directory now I'm going to set the top two here for the driver and for the database string and for this video I'm just going to do this on the command line using the export command so we're going to export the Gus driver environment variable and we're going to set that to postgress and the second one was the Gus database string we're going to set that to what we were using in the goose up command and once we've set those what we can do is we can just run Goose up without having to worry about those and you can see that it works but it doesn't have any migrations to run against that database if I was to run the ghost down command it was going to revert those previous migrations but notice that we no longer have to pass that information we can just run these simple commands and Gus will know what database to connect to and it will know what driver to use and again if we go back to the postgres Explorer here now that we've run at the ghost down command if I refresh the public schema you can see it's removed that coach table and in the team table we're back to having a VAR car field here for the coach now if I run G down again it's essentially going to remove that databased t for the teams and when we run the Gus up command what it's going to do and I'm going to clear the terminal here is it's going to apply all of the migrations in the correct order so we now have two migrations it's going to apply the first one and then it will apply the second to get the database to the latest state but as we saw earlier you can actually run different commands here so if you don't want to run all of the migrations you can specify different commands for that one of them is the up by one command and what that's going to do is it's going to take the current state and it's going to apply the next migration and that can be useful if you want to incrementally step through the different migrations so you can see that when we run this command it only runs one of the migrations and that's for adding the team table and again when we refresh the schema you can see that table appearing on the left hand side and if we run that one more time it's going to apply that second migration and that's going to add that coach table as you can see here so this gives you a

way to step through the migrations and you can also do this when you're running ghost down now I want to create one other migration just to show that you can actually create test data as well so let's run Goose create here and we'll give this one a name of test data and that's going to be an SQL migration and that will add that file that we can now open on the left hand side and once we've opened that what I'm going to do when we run the Gus up command is Define some test data to be added to the database so we're going to add a coach here and let's add one called Pep Guardiola and then we can add a team here of Manchester City and we're going to set the city and the coach ID remember it's a foreign key so we specify the ID for that coach now the ghost down command is going to be quite simple we're just going to delete everything from the team and the coach table and we need to do it in that order because of the foreign keys so if we go back to the command line here and we rerun Gus up that's going to add that new data into the database and if we go to the postgres Explorer here I'm going to refresh what we have here and let's look at the team table so if we select the top 1,000 records you can see we now have Manchester City appearing in this table and similarly for the coach table we can run a select and we can see the data that's been added here now I'm back in main.go here and I'm going to paste some code in just to finish this video what we're going to do is we're going to actually query the database so we've made those changes via the migration files and now we're going to write some go code to actually connect to the database and to pull that data out and just display it in the terminal for this video now in order to do that I've installed the PQ driver for post so we've imported that and if we go to the main function we Define our connection string and we pass that into the SQL do open function we then check to make sure we don't have an error and we defer the db. close method and you can see the comments here just explaining what we're doing so we're going to test the connection using db. ping and make sure that we don't have any kind of error and then we can print to the terminal that we've successfully connected to the database we can then query the database for the teams and we can use the db. query function for that and we pass in an SQL string here that we want to actually execute against the database so we're selecting those columns from the team table and then we check if we have an error and we defer the row. close method so we get back some rows from db. query and we can iterate over those rows here as you can see in this for Loop so when we call rows. next it's going to give us the next row and what we can do here is scan the results from that row and we're scanning these into some variables that we've set up on the two lines above so two integers and two strings and then we can print these out to the terminal so let's just run this and make sure it works and that will be all for this video we can use the go run command and it's main.go and when we run that we get the message successfully connected to the database and we have a list of the teams in this case there's only a single team and that's Manchester City but we get that information coming from the database and what is important for this video is that we've used gust to actually manage the schema in the database and manage how that changes and evolves over time now I'm going to use G in an upcoming course that I'm creating on using go with HTM X and Temple and that's going to be fully integrated with a database but if you have any other ideas for for goal content on YouTube let me know in the comments as I said at the start of the video some of the Go videos have been extremely popular and I'd love to make more because I love this language and it's something I like to use outside of work and try and learn as much as possible so thanks again for watching this video if you have enjoyed it check out the coffee page that we've got in the description and give the video a thumbs up if you found this content useful as well that would be amazing and we'll see you in the next video