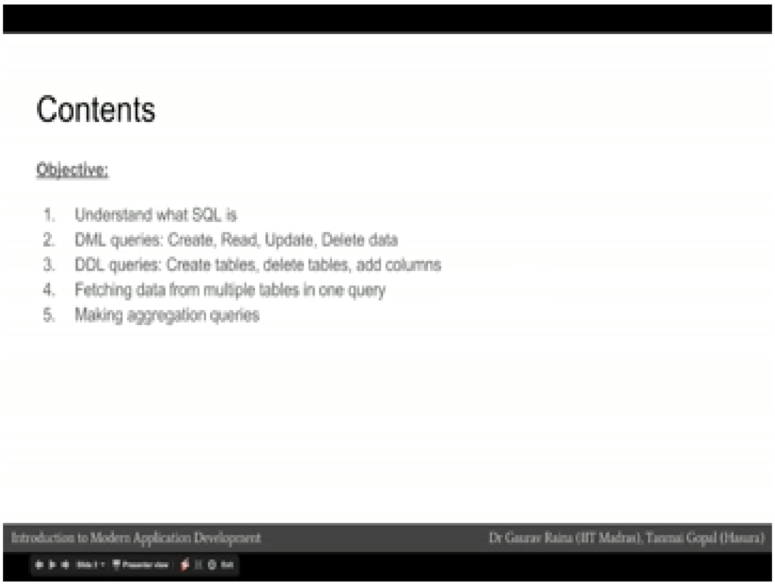
Introduction to Modern Application Development
Dr. Gaurav Raina
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - P9
Lecture - 17
Introduction to SQL

Hi all. Welcome to Module - P9. In this module, we will be getting an introduction to what SQL is, and doing some quick exercises with SQL.

(Refer Slide Time: 00:12)



Contents

Objective:

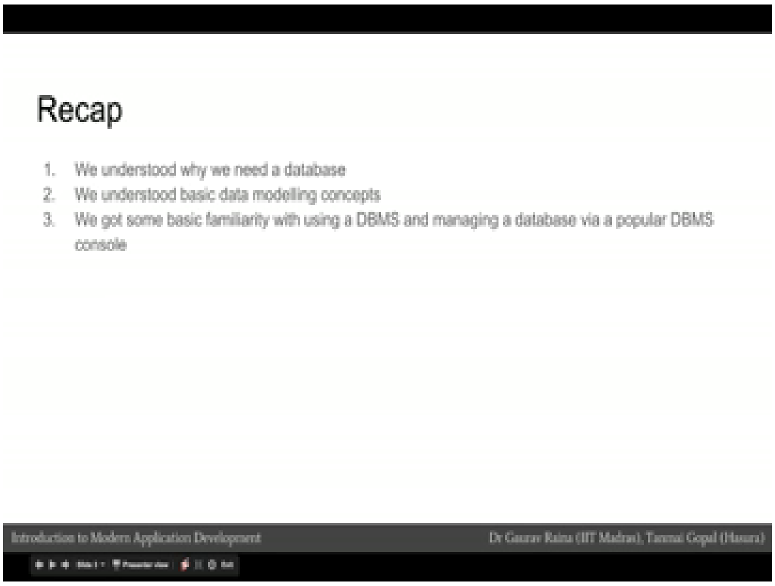
1. Understand what SQL is
2. DML queries: Create, Read, Update, Delete data
3. DDL queries: Create tables, delete tables, add columns
4. Fetching data from multiple tables in one query
5. Making aggregation queries

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

Main objective is to understand what SQL is and what kind of queries we can do with it, and we will make several different kinds of queries.

(Refer Slide Time: 00:18)



Recap

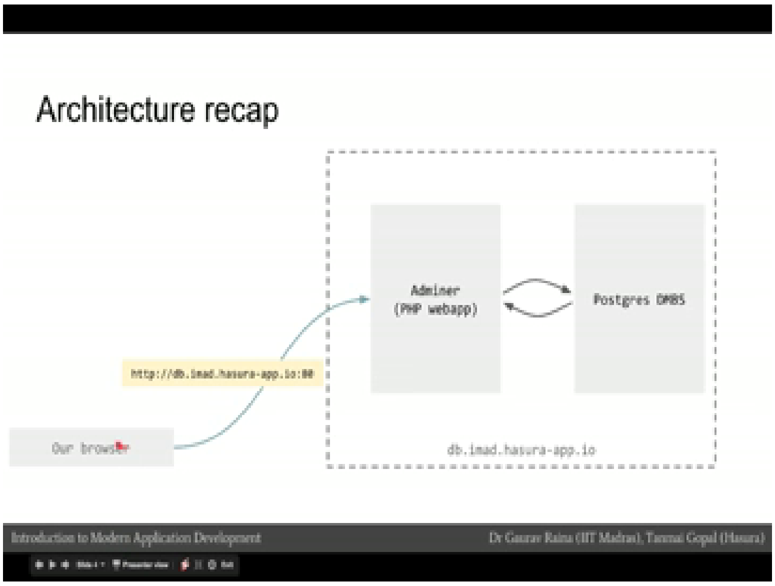
1. We understood why we need a database
2. We understood basic data modelling concepts
3. We got some basic familiarity with using a DBMS and managing a database via a popular DBMS console

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

To recap what we have understood so far. We know why we need a database. We have also understood some basic data modeling concepts. We have gotten some basic familiarity with using a DBMS, and also managing a database.

(Refer Slide Time: 00:32)



Architecture recap

Adminer (PHP webapp)

Postgres DMBS

http://db.imad.hasura-app.io:80

Our browser

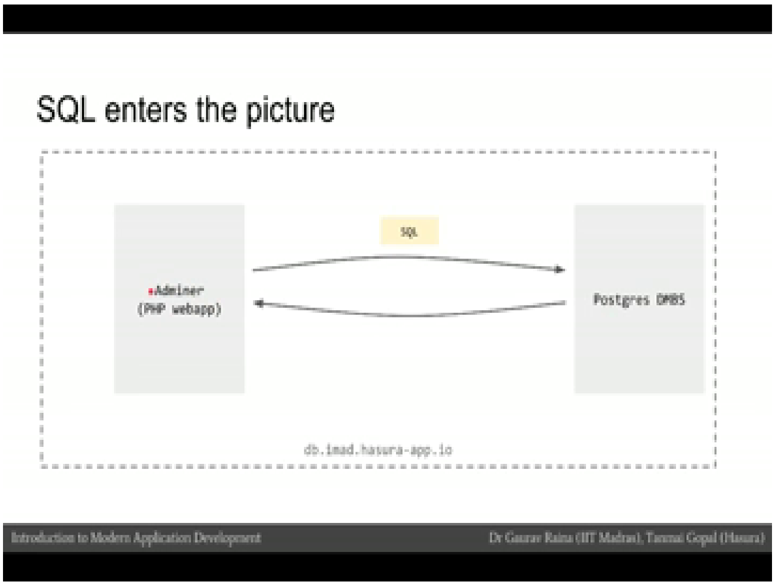
db.imad.hasura-app.io

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

This is what the architecture look like. We had a browser; using the browser, we went to a webapp, which was db.imad.hasura.app.io, which is serving an application called Adminer. Now this application was a PHP webapp, which was talking to the Postgres DBMS; taking results from the Postgres DBMS, and sending it back to us.

(Refer Slide Time: 00:51)



SQL enters the picture

SQL

Adminer
(PHP webapp)

Postgres DBMS

db.imad.hasura-app.io

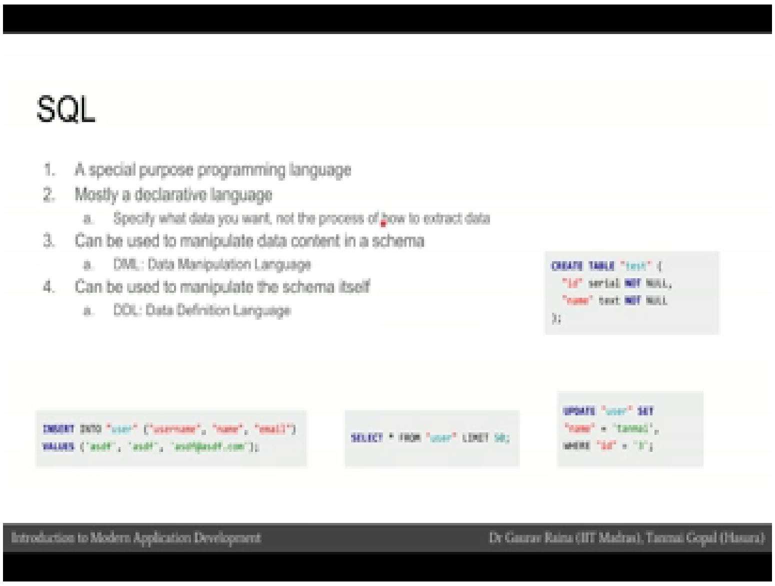
Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

If we look at how Adminer which is the webapp is communicating with Postgres, Adminer is a webapp which is actually a client and the DBMS is the server. So, the Adminer client makes a request to the DBMS server and the DBMS server returns the response the Adminer client. This communication happens over a different network protocol, which is not http.

For Postgres, in fact, it is call the Postgres protocol. It is the protocol on top of the TCP protocol, like how http is a protocol on top of the TCP protocol. Adminer sends an SQL query to the Postgres DBMS; and the Postgres DBMS responds to the results. So, the language in which the webapp is making queries to the DBMS is SQL. This is similar to how when we make http request. We make http request that have a particular format, for example, we make a get request to the URL, our request contains the URL, it contains the path, it contains headers, and it contains a body. Similarly, the request that is sent from the Adminer to the DBMS contains SQL.

(Refer Slide Time: 01:59)



SQL

1. A special purpose programming language
2. Mostly a declarative language
   a. Specify what data you want, not the process of how to extract data
3. Can be used to manipulate data content in a schema
   a. DML: Data Manipulation Language
4. Can be used to manipulate the schema itself
   a. DDL: Data Definition Language

```
CREATE TABLE "test" (
  "id" serial NOT NULL,
  "name" text NOT NULL
);
```

```
INSERT INTO "user" ("username", "name", "email")
VALUES ('asdf', 'asdf', 'asdf@asdf.com');
```

```
SELECT * FROM "user" LIMIT 50;
```

```
UPDATE "user" SET
"name" = 'tanmai',
WHERE "id" = '1';
```

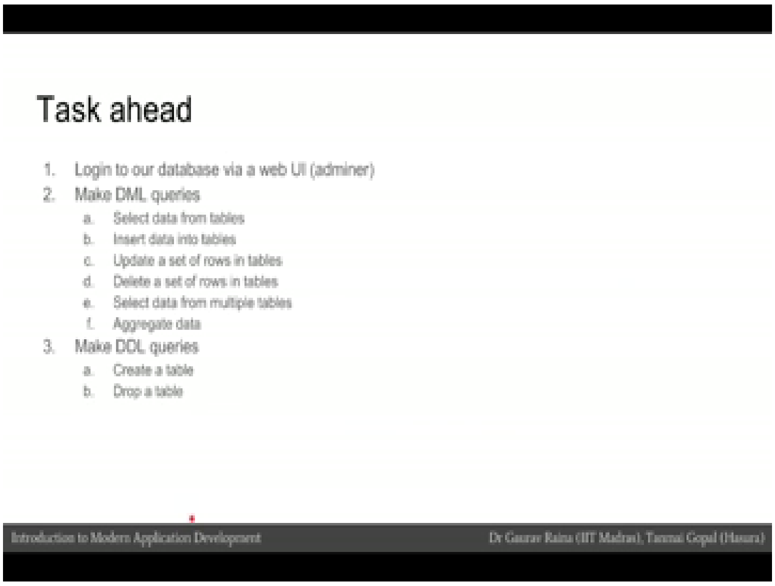
Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

Now, what is SQL? Now SQL is a special purpose language; it is a declarative language; it is not a procedural language. So, with SQL, we can specify what we want to do, as a result and we can declare our action, but we do not need to describe how the action needs to be done. We will understand what this means in a little bit, and you will get sense of this.

The two quick definitions that are important; SQL is both DML and DDL. DDM is a data manipulation language; so it is the language to manipulate data meaning to select, insert, update, or delete data. A DDL is a data definition language, which means a language to define the schema or the structure in which the data is stored, and to change that structure in which data is stored. So, SQL allows us to do both; it allow us to for example, in this box, insert a value into a table or select the value from a table, but it also allows us to create a table.

(Refer Slide Time: 03:03)



Task ahead

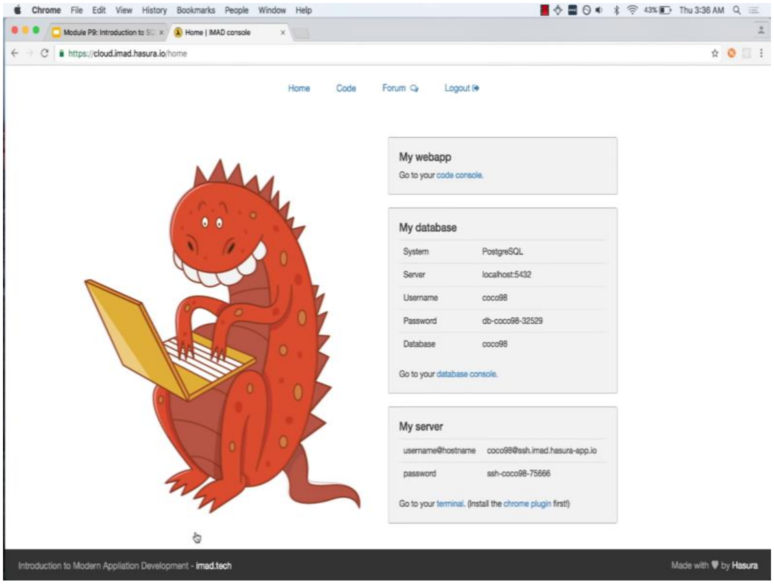
1. Login to our database via a web UI (adminer)
2. Make DML queries
   a. Select data from tables
   b. Insert data into tables
   c. Update a set of rows in tables
   d. Delete a set of rows in tables
   e. Select data from multiple tables
   f. Aggregate data
3. Make DDL queries
   a. Create a table
   b. Drop a table

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

So, let us quickly try this out.

(Refer Slide Time: 03:08)



Chrome File Edit View History Bookmarks People Window Help
Thu 3:36 AM
Module P9: Introduction to SQ
Home | IMAD console
https://cloud.imad.hasura.io/home
Home
Code
Forum
Logout

My webapp

Go to your code console.

My database

System	PostgreSQL
Server	localhost:5432
Username	coco98
Password	db-coco98-32529
Database	coco98

Go to your database console.

My server

username@hostname	coco98@ssh.imad.hasura-app.io
password	ssh-coco98-75666

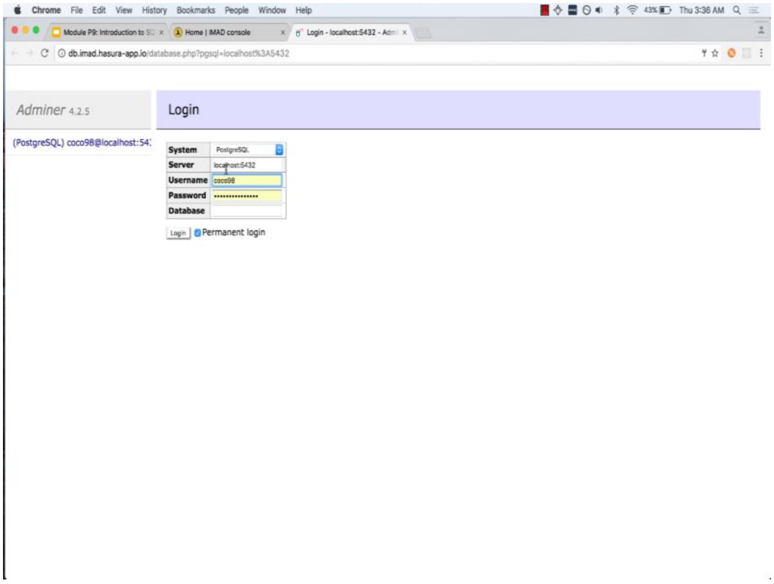
Go to your terminal. (Install the chrome plugin first!)

Introduction to Modern Appliation Development - imad.tech

Made with ♥ by Hasura

I head to my IMAD console; go to the database console.

(Refer Slide Time: 03:11)



db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432

Adminer 4.2.5

Login

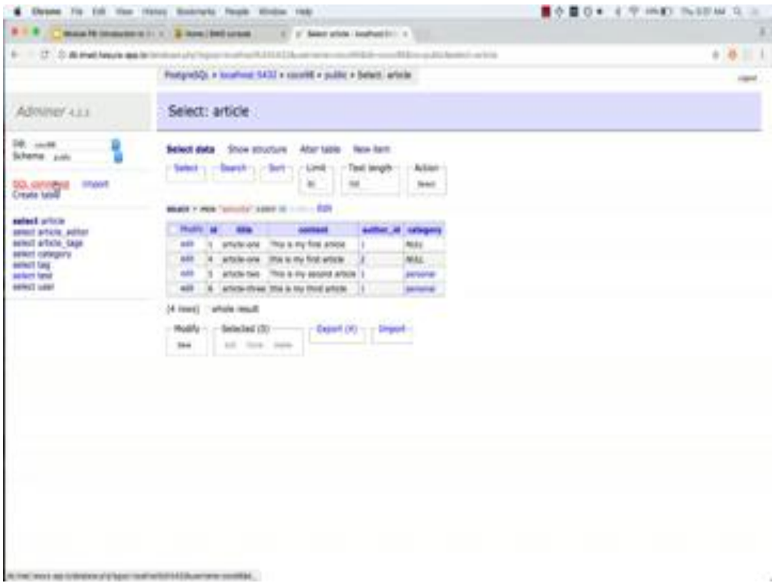
(PostgreSQL) coco98@localhost:54

System	PostgreSQL
Server	localhost:5432
Username	coco98
Password	••••••••••••••••
Database	

Login ☑ Permanent login

I login here.

(Refer Slide Time: 03:19)



PostgreSQL » localhost:5432 » coco98 » public » Select: article

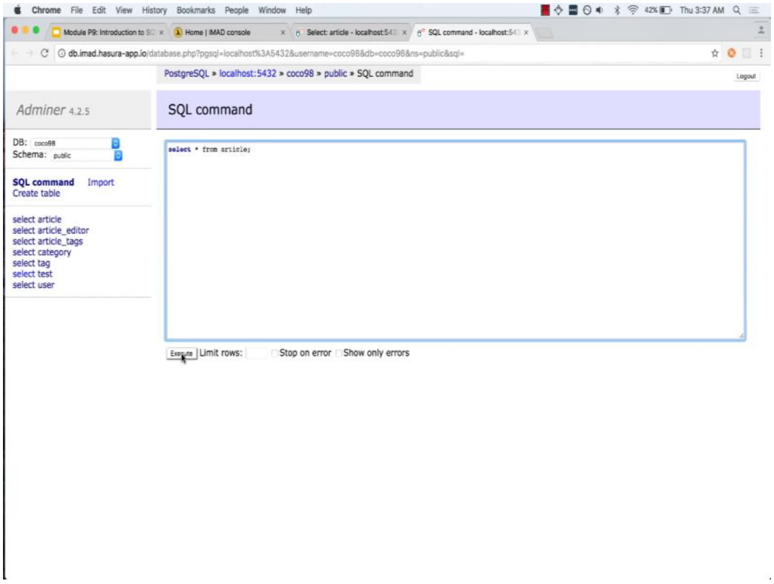
Select: article

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	personal

(4 rows)

Let us look at the article table. So, we have 4 articles in the article table.

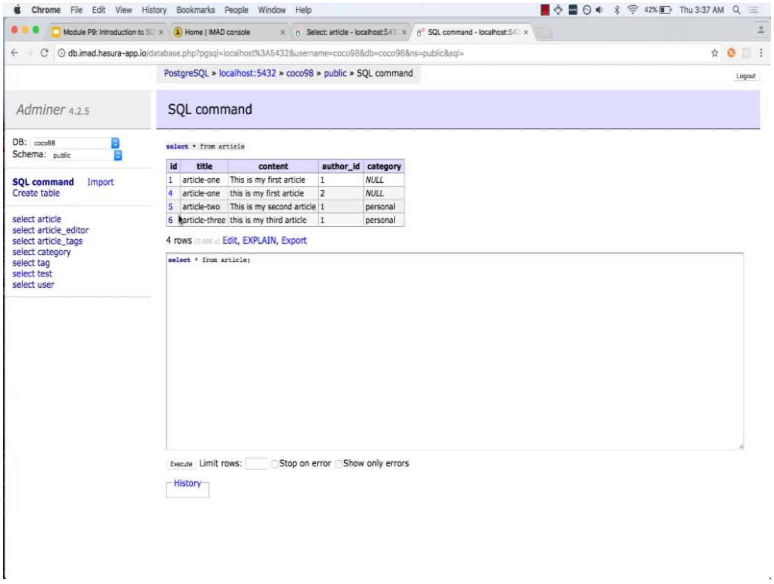
(Refer Slide Time: 03:28)



```
select * from article;
```

I am going to open this in a separate tab. So, the SQL command window I will open in a separate tab. And now instead of using the UI to select this data let us try it use the SQL to select the data. So, I will say select star from article and let see what it does.

(Refer Slide Time: 03:43)



```
select * from article
```

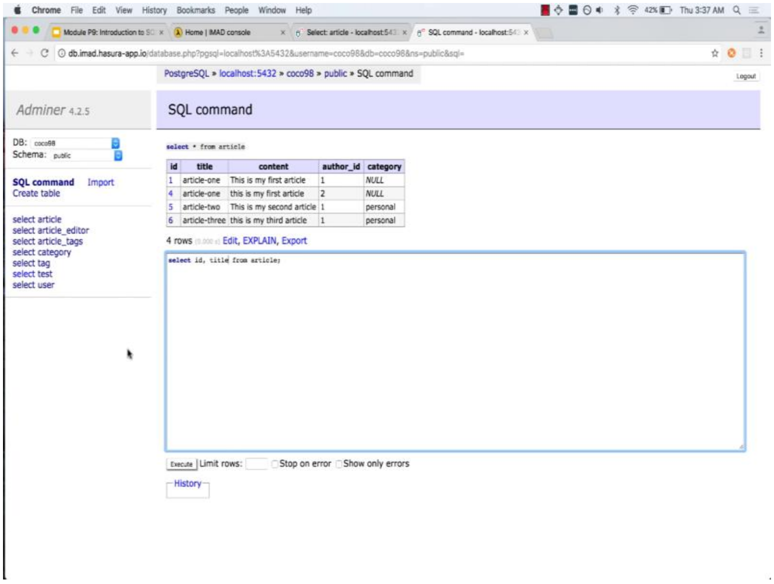
id	title	content	author_id	category
1	article-one	This is my first article	1	NULL
4	article-one	this is my first article	2	NULL
5	article-two	This is my second article	1	personal
6	article-three	this is my third article	1	personal

4 rows Edit, EXPLAIN, Export

```
select * from article;
```

This selects the 4 articles that we have. So, let read this SQL query we are saying select which is an SQL keywords. So, we are seeing select star which represents all columns. So, select all the columns from which again it is an SQL keyword from the table article.

(Refer Slide Time: 04:04)



SQL command

```
select * from article
```

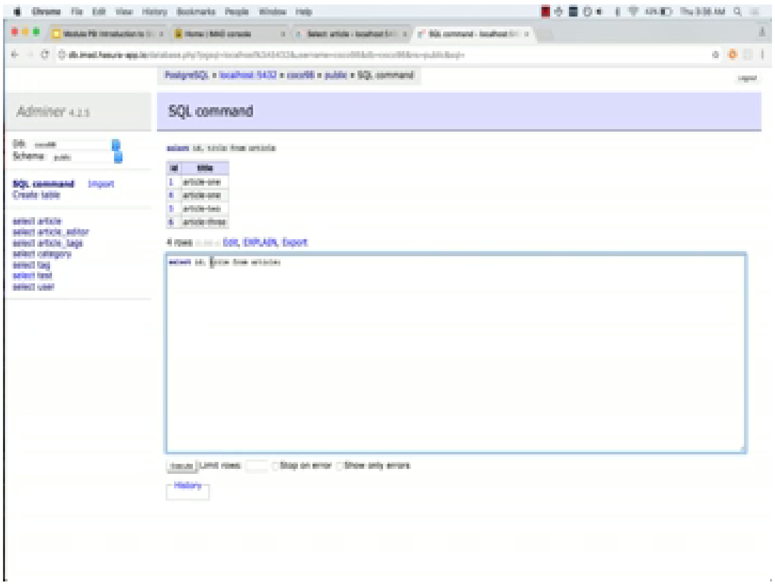
id	title	content	author_id	category
1	article-one	This is my first article	1	NULL
4	article-one	this is my first article	2	NULL
5	article-two	This is my second article	1	personal
6	article-three	this is my third article	1	personal

4 rows Edit, EXPLAIN, Export

```
select id, title from article;
```

I can make this query a little better, and I can change select id coma title from the articles. So, we will just select two columns from the article.

(Refer Slide Time: 04:09)



SQL command

```
select id, title from article
```

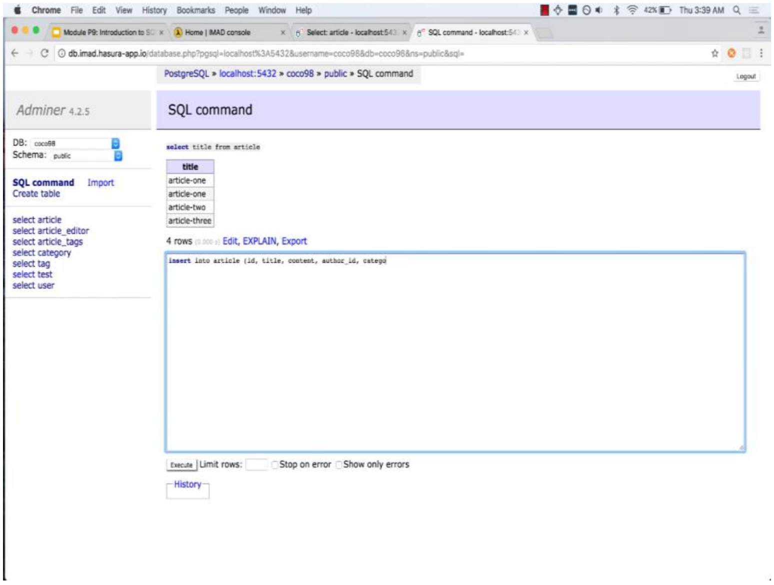
id	title
1	article-one
4	article-one
5	article-two
6	article-three

4 rows Edit, EXPLAIN, Export

```
select id, title from article;
```

And try to I can select also only the title. So, each of this is a SQL query that we made and it returns the results. Though as you can see the results that I returned by an SQL query resembles a table and is exactly the same at a logical level is exactly the same as a table. So, the output is also in a tabular format.

(Refer Slide Time: 04:46)


Chrome File Edit View History Bookmarks People Window Help
Thu 3:39 AM
Module P9: Introduction to SQ
Home | IMAD console
Select: article - localhost:543
SQL command - localhost:543
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&sql=
PostgreSQL » localhost:5432 » coco98 » public » SQL command
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select article_editor
select article_tags
select category
select tag
select test
select user

SQL command

```
select title from article
```

title
article-one
article-one
article-two
article-three

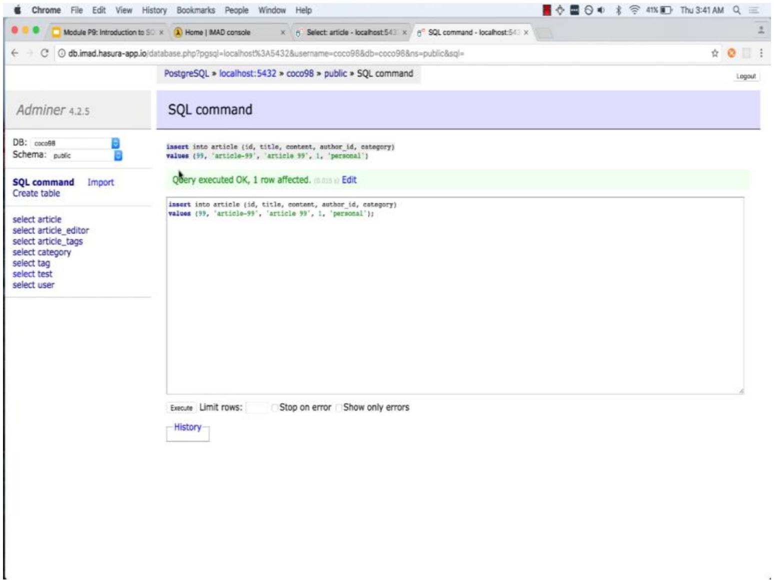
4 rows Edit, EXPLAIN, Export

```
insert into article (id, title, content, author_id, catego
```

Execute
Limit rows:
□ Stop on error
□ Show only errors
History

Let us try to now make an insert request. So, this is what an insert request looks like. I will say insert into and then I will give you the table names, so table name article. I then have to give the columns that I am inserting I have to name the columns that I am trying to insert into. Let us check out what the columns are. So, the columns are id, title, content, author id and category – so id, title, content, author id, and category.

(Refer Slide Time: 05:14)



PostgreSQL » localhost:5432 » coco98 » public » SQL command

SQL command

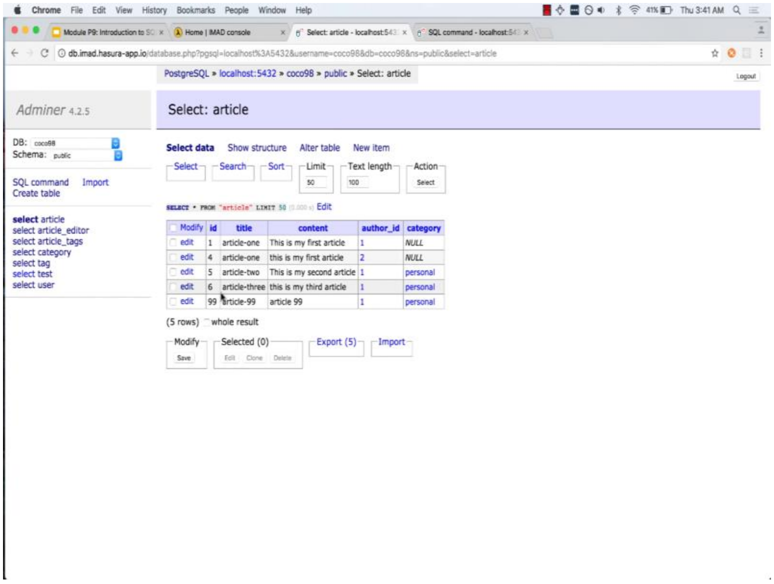
```
insert into article (id, title, content, author_id, category)
values (99, 'article-99', 'article 99', 1, 'personal')
```

Query executed OK, 1 row affected. Edit

```
insert into article (id, title, content, author_id, category)
values (99, 'article-99', 'article 99', 1, 'personal');
```

And I am inserting into this article table the values, and now I am going to list out all the values of this particular row. So, I am going to create a new id, let say 99, the title of my article is article 99. The content is also article 99. The author id is 1 which is in integer, and the category is say a personal. And once I execute this query you can see that I am inserting into the article table.

(Refer Slide Time: 05:49)



PostgreSQL » localhost:5432 » coco98 » public » Select: article

Select: article

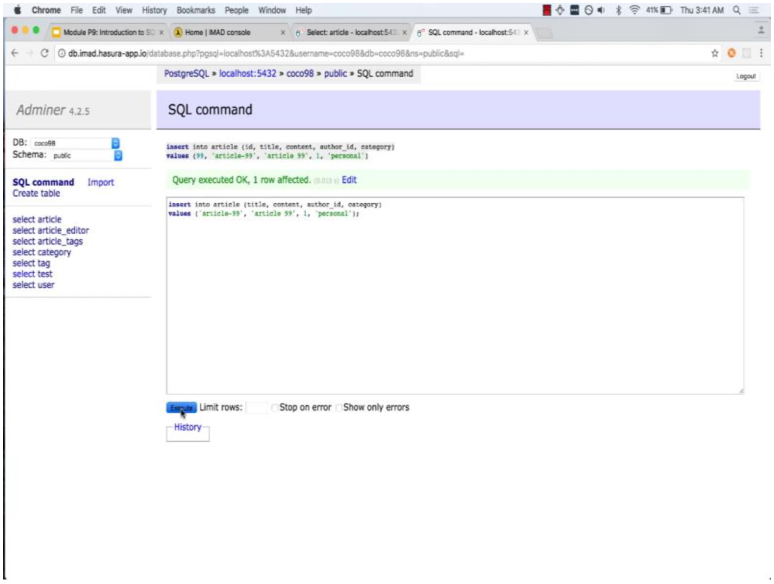
```
SELECT * FROM "article" LIMIT 50
```

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	personal
edit	99	article-99	article 99	1	personal

(5 rows)

And let us refresh this to see, if they are insert has gone through we can see that a new row has been created.

(Refer Slide Time: 06:02)



SQL command

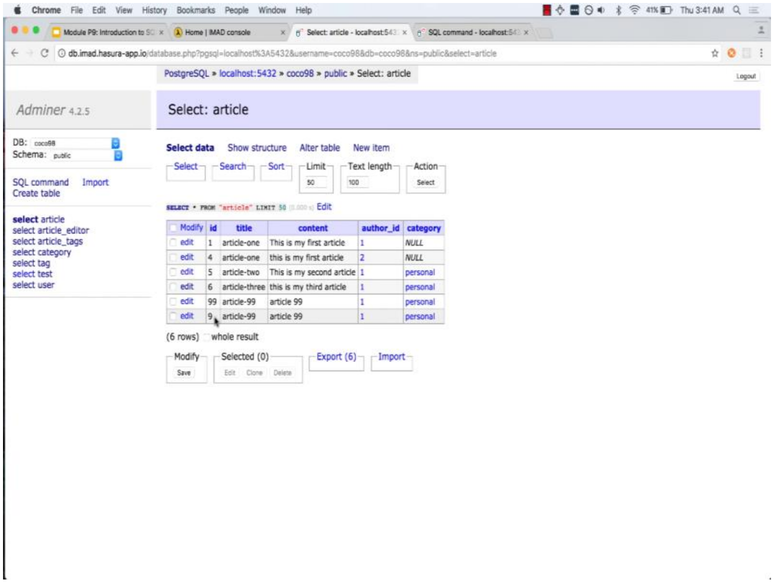
```
insert into article (id, title, content, author_id, category)
values (99, 'article-99', 'article 99', 1, 'personal')
```

Query executed OK, 1 row affected. Edit

```
insert into article (title, content, author_id, category)
values ('article-99', 'article 99', 1, 'personal');
```

Now, the interesting thing is that we actually do not need to insert an id value. So, because the id is set to auto increment, so I can remove the id column and I can remove the id value from here.

(Refer Slide Time: 06:12)



PostgreSQL » localhost:5432 » coco98 » public » Select: article

Select: article

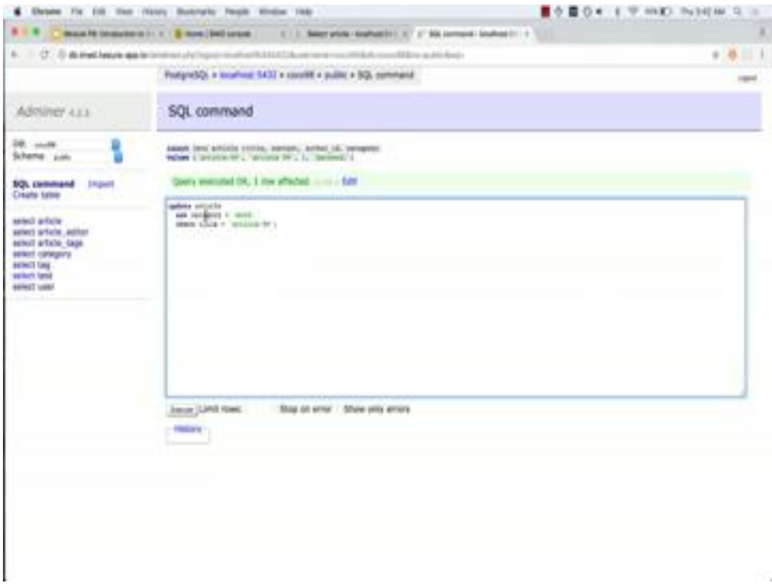
```
SELECT * FROM "article" LIMIT 50
```

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	personal
edit	99	article-99	article 99	1	personal
edit	9	article-99	article 99	1	personal

(6 rows)

Let us I inserting again and you can see that another article has been created, but it has been given the id 9, because that is what a last auto incrementing number was at.

(Refer Slide Time: 06:23)



PostgreSQL » localhost:5432 » coco98 » public » SQL command

SQL command

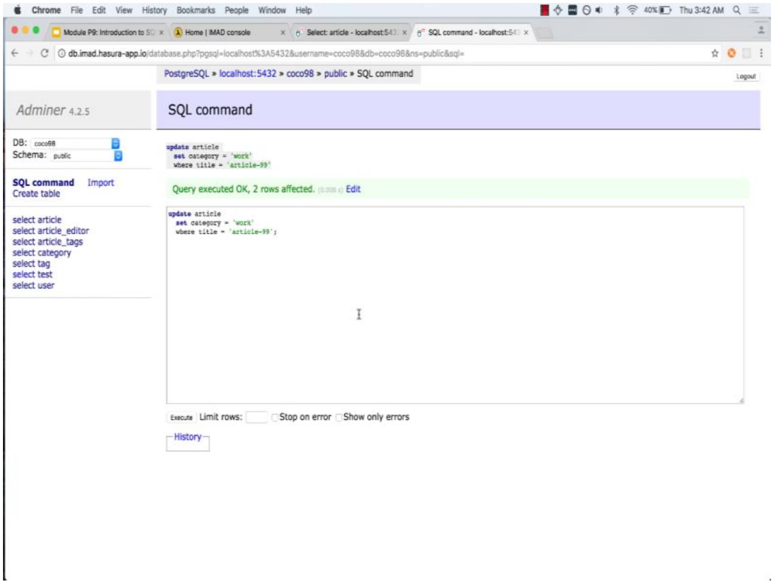
```
insert into article (title, content, author_id, category)
values ('article-99', 'article 99', 1, 'personal');
```

Query executed OK, 1 row affected.

```
update article
  set category = 'work'
  where title = 'article-99';
```

Let us now look at a different request let say let us look at and update a request. Let us do update article, set category equal to work, where title equal to article 99. In this query, I am trying to update the article table; in the article table, I am trying to set the value of category to work and I want this to happen for all the rows in the article table only where a particular condition is met, which is where the title should be equal to 99. Otherwise, the article should not be updated.

(Refer Slide Time: 07:00)



Chrome File Edit View History Bookmarks People Window Help
Thu 3:42 AM
Module P9: Introduction to SQ
Home | IMAD console
Select: article - localhost:543
SQL command - localhost:54
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&sql=
PostgreSQL » localhost:5432 » coco98 » public » SQL command
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select article_editor
select article_tags
select category
select tag
select test
select user
Execute
History

SQL command

```
update article
  set category = 'work'
  where title = 'article-99'
```

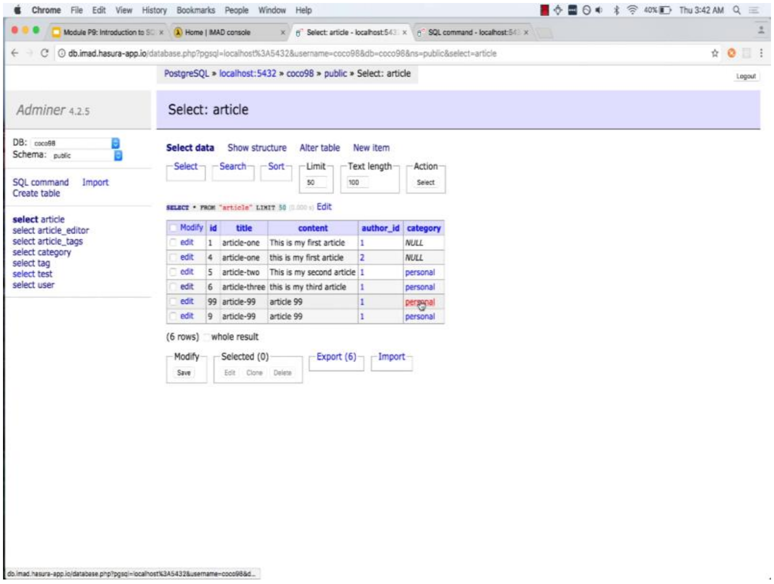
Query executed OK, 2 rows affected. Edit

```
update article
  set category = 'work'
  where title = 'article-99';
```

Limit rows: □ Stop on error □ Show only errors

Let us try to execute this query see that it says query executed two rows affected because I had two rows of the same title.

(Refer Slide Time: 07:08)



DB: coco98
Schema: public

Limit: 50
Text length: 100

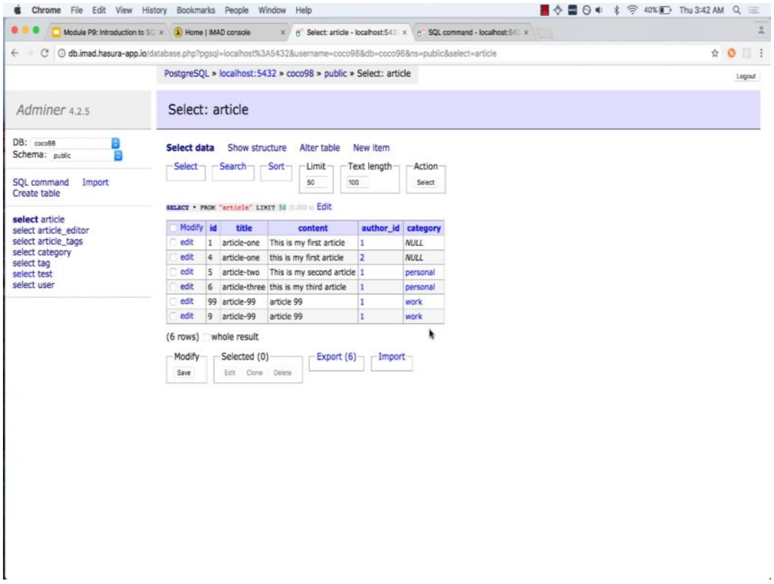
```
SELECT * FROM "article" LIMIT 50
```

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	personal
edit	99	article-99	article 99	1	personal
edit	9	article-99	article 99	1	personal

(6 rows) whole result

Now, if I look at the old data it says the category is personal,

(Refer Slide Time: 07:11)



DB: coco98
Schema: public

Limit: 50
Text length: 100

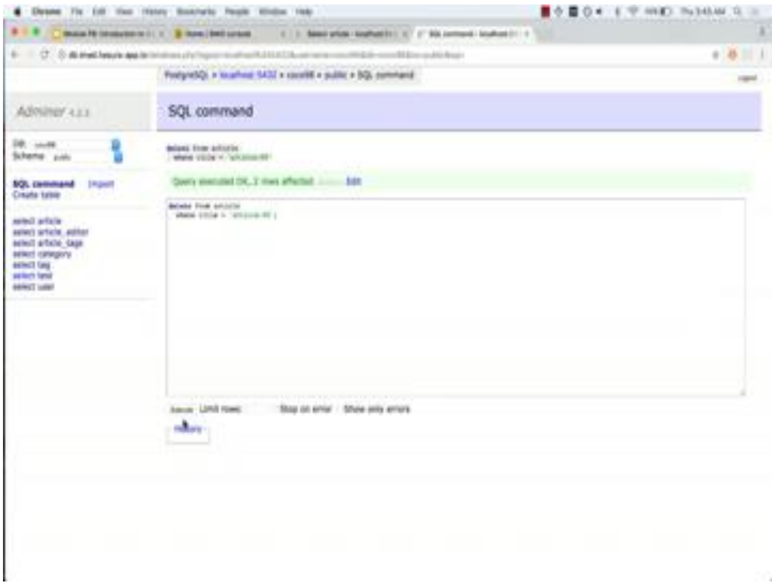
```
SELECT * FROM "article" LIMIT 50
```

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	personal
edit	99	article-99	article 99	1	work
edit	9	article-99	article 99	1	work

(6 rows) whole result

If I refresh this, you can see that the article is changed to work.

(Refer Slide Time: 07:16)



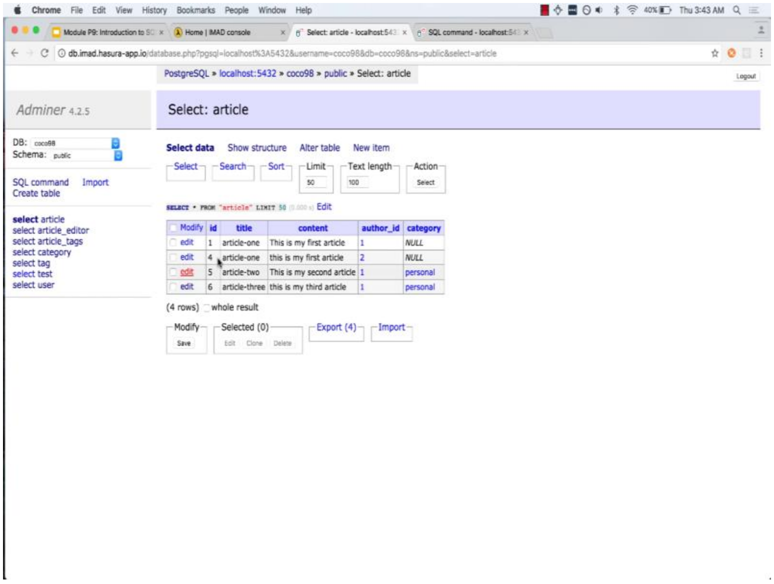
```
delete from article
 where title = 'article-99'
```

Query executed OK, 2 rows affected.

```
delete from article
 where title = 'article-99';
```

I can now also issue a delete request. Let us do delete from article where title equal to article 99.

(Refer Slide Time: 07:29)



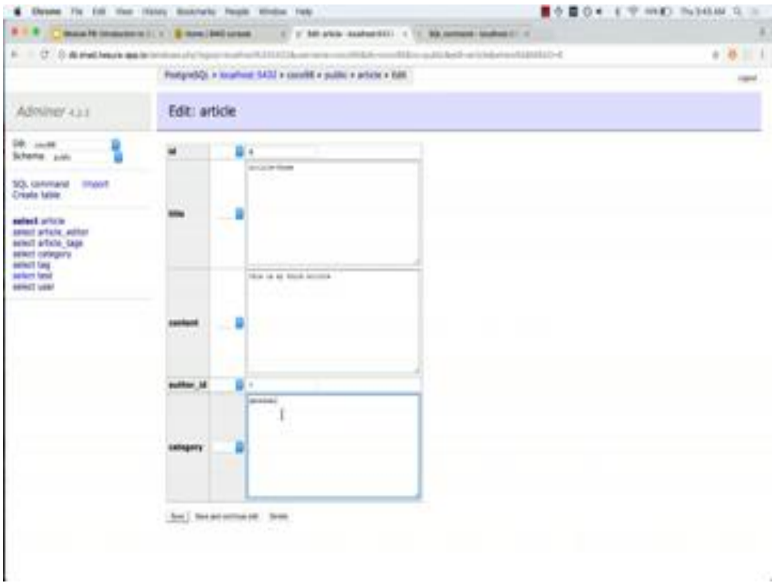
```
SELECT * FROM "article" LIMIT 50
```

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	personal

(4 rows)

So, again is has two rows affecting first look at this and you can see that both those rows are gone now. In fact, every single action that you can do on this UI on top of the database is actually done an SQL command. So, an easy way to see that it is actually by looking at this particular part of the Adminer console which shows us the SQL query.

(Refer Slide Time: 07:57)

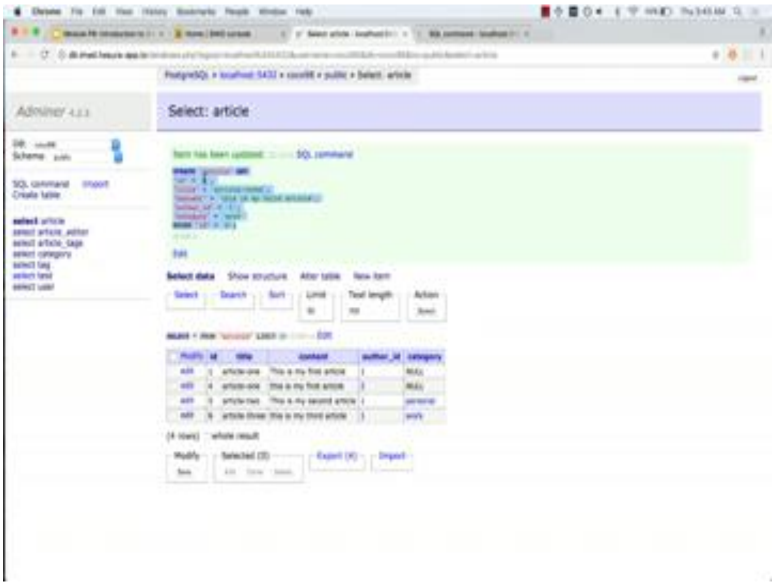


Edit: article

id	
title	article-three
content	this is my third article
author_id	
category	personal

For example, if I added this and changed the category to work, you can see that the item has been updated.

(Refer Slide Time: 08:07)



Select: article

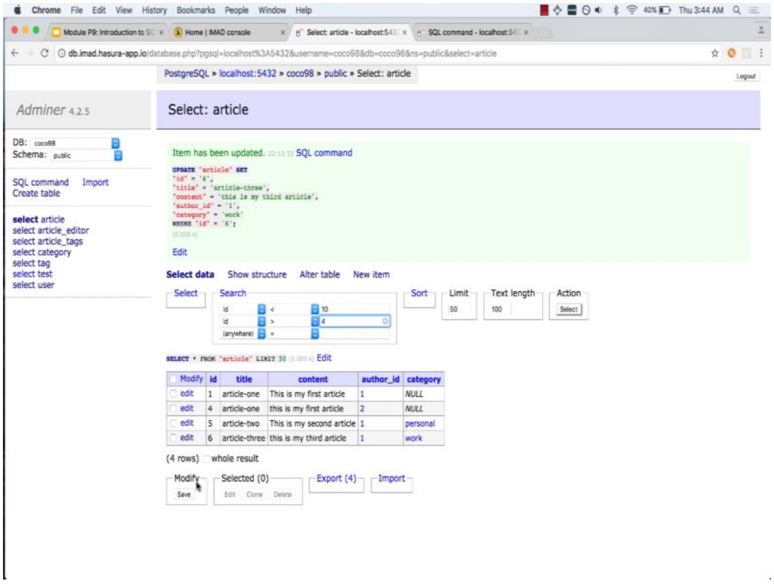
Item has been updated.

Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	1	NULL
edit	3	article-two	This is my second article	1	personal
edit	6	article three	this is my third article	1	work

(4 rows) whole result

You can click on the SQL command; you will see the SQL command that the web app is issuing to the d b to the DBMS right. So, you can learn SQL basics just by making requests here.

(Refer Slide Time: 08:26)


Chrome File Edit View History Bookmarks People Window Help
Thu 3:44 AM
Module P9: Introduction to SQ
Home | IMAD console
Select: article - localhost:543
SQL command - localhost:543
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article
PostgreSQL » localhost:5432 » coco98 » public » Select: article
Logout
Adminer 4.2.5
SQL command
Import
Create table
select article
select article_editor
select article_tags
select category
select tag
select test
select user
Select data
Show structure
Alter table
New item
Select
Sort
Edit
Save
Clone
Delete
Export (4)
Import

Select: article

DB: coco98
Schema: public

Item has been updated. 22:13:33 SQL command

```
UPDATE "article" SET
"id" = '6',
"title" = 'article-three',
"content" = 'this is my third article',
"author_id" = '1',
"category" = 'work'
WHERE "id" = '6';
(0.006 s)
```

Edit

Search
id < 10
id > 4
(anywhere) =

Limit 50
Text length 100
Action Select

```
SELECT * FROM "article" LIMIT 50 (0.000 s) Edit
```

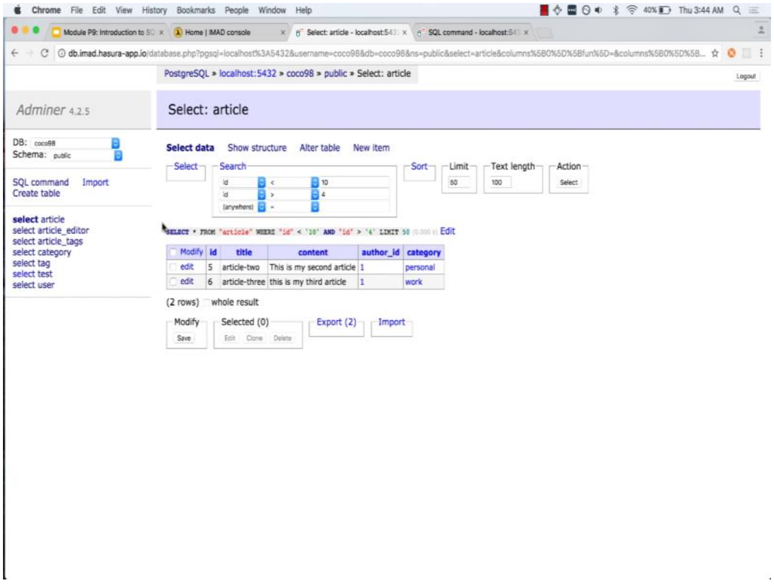
Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	work

(4 rows) whole result

Modify
Selected (0)

For example, if you want to select by a particular values, so let say I want to select where id less than 10, and id greater than 4, right, let say we make this query.

(Refer Slide Time: 08:32)



Chrome File Edit View History Bookmarks People Window Help
Thu 3:44 AM
Module P9: Introduction to SQ
Home | IMAD console
Select: article - localhost:543
SQL command - localhost:54
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article&columns%5B0%5D%5Bfun%5D=&columns%5B0%5D%5B...
PostgreSQL » localhost:5432 » coco98 » public » Select: article
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command Import
Create table
select article
select article_editor
select article_tags
select category
select tag
select test
select user
Select data Show structure Alter table New item
Select
Search
Sort
Limit
Text length
Action
Select
Modify
Save
Selected (0)
Edit Clone Delete
Export (2)
Import

Select: article

id	<	10
id	>	4
(anywhere)	=	

Limit: 50 Text length: 100

```
SELECT * FROM "article" WHERE "id" < '10' AND "id" > '4' LIMIT 50
```

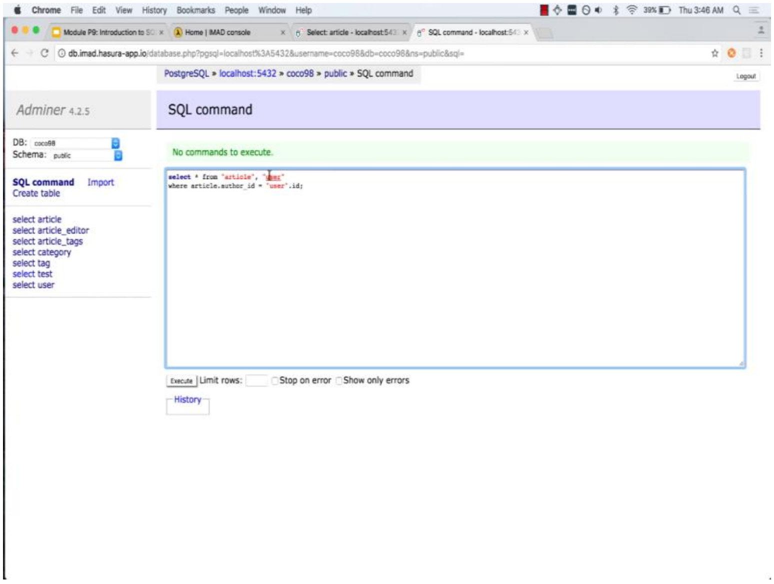
Modify	id	title	content	author_id	category
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	work

(2 rows) whole result

You get two values and you can see the SQL statement that is going to make this query, so that is the SQL statement. Let us now look at a slightly different kind of query let us look at a query where we can fetch information from multiple tables in this same request and that is a very common use case.

For example, if you look at the article table, if I fetch data from the article table I will only get id, title, content, author id and category, but suppose I also want the authors name how will I make that query. I do not want to make making two quarries is a bad idea; I do not want to make query to the article table and then make a query to the author table and fetch that information. Because if the data set is large, for example, if you imagine a few 1000 articles I would have to make one query to fetch the 1000 articles and then, I would have to make potentially up to a 1000 more queries to fetch the author information for each author. But I want to fetch all of that information in one short, and so let see where that query looks like.

(Refer Slide Time: 09:30)



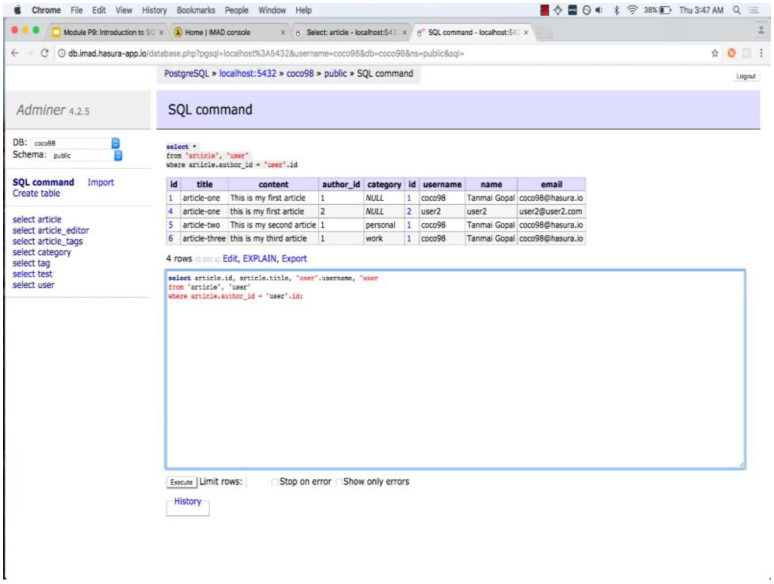
SQL command

No commands to execute.

```
select * from "article", "user"
where article.author_id = "user".id;
```

Here I can say select a star from article comma user, where article dot author id is equal to user dot id. Note that is important to used the double quotes around the user table name because user is a special keyword in Postgres, and so it is important to add the double quotes here. So, if you say select star from. So, I am saying select all the columns from both these tables where article dot author id is equal to user dot id.

(Refer Slide Time: 10:06)



SQL command

```
select *
from "article", "user"
where article.author_id = "user".id
```

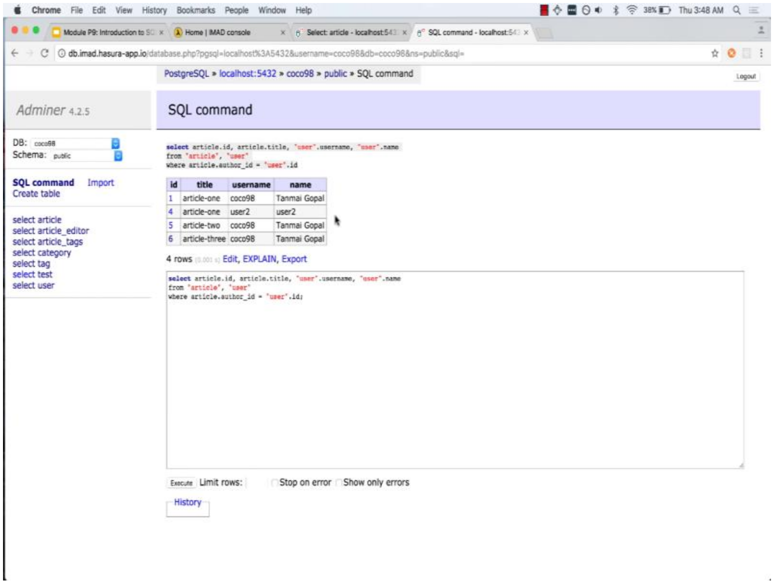
id	title	content	author_id	category	id	username	name	email
1	article-one	This is my first article	1	NULL	1	coco98	Tanmai Gopal	coco98@hasura.io
4	article-one	this is my first article	2	NULL	2	user2	user2	user2@user2.com
5	article-two	This is my second article	1	personal	1	coco98	Tanmai Gopal	coco98@hasura.io
6	article-three	this is my third article	1	work	1	coco98	Tanmai Gopal	coco98@hasura.io

4 rows (0.001 s) Edit, EXPLAIN, Export

```
select article.id, article.title, "user".username, "user
from "article", "user"
where article.author_id = "user".id;
```

And so if you execute this, you will get all the author information here as well right. So, getting author id 1, you are getting user name, name, and email. And so, now you can choose what columns you want. For example, you want article dot id article dot title and then I want the user dot user name user dot name.

(Refer Slide Time: 10:39)



```
select article.id, article.title, "user".username, "user".name
from "article", "user"
where article.author_id = "user".id
```

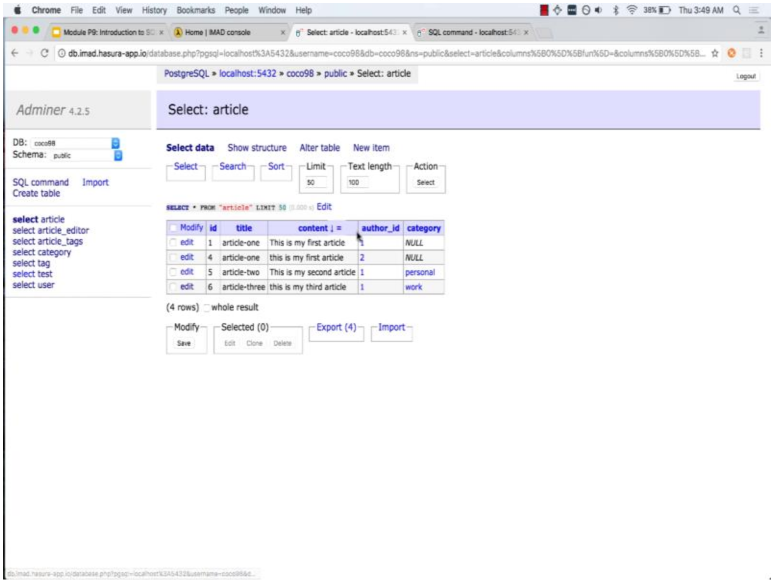
id	title	username	name
1	article-one	coco98	Tanmai Gopal
4	article-one	user2	user2
5	article-two	coco98	Tanmai Gopal
6	article-three	coco98	Tanmai Gopal

4 rows (0.001 s) Edit, EXPLAIN, Export

```
select article.id, article.title, "user".username, "user".name
from "article", "user"
where article.author_id = "user".id;
```

And if these are the columns that you want you can make a query, and you can just get those 4 columns. This is how you can fetch from multiple tables in the same short. Now, this is a very simple example and what we will actually doing here is something called a joint. So, this is a shortcut for doing something called a join. Joints allow you to fetch from multiple tables across several conditions.

(Refer Slide Time: 11:11)



db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article&columns%5B0%5D%5Bfun%5D=&columns%5B0%5D%5B...

PostgreSQL » localhost:5432 » coco98 » public » Select: article

Select: article

DB: coco98
Schema: public

Limit: 50
Text length: 100

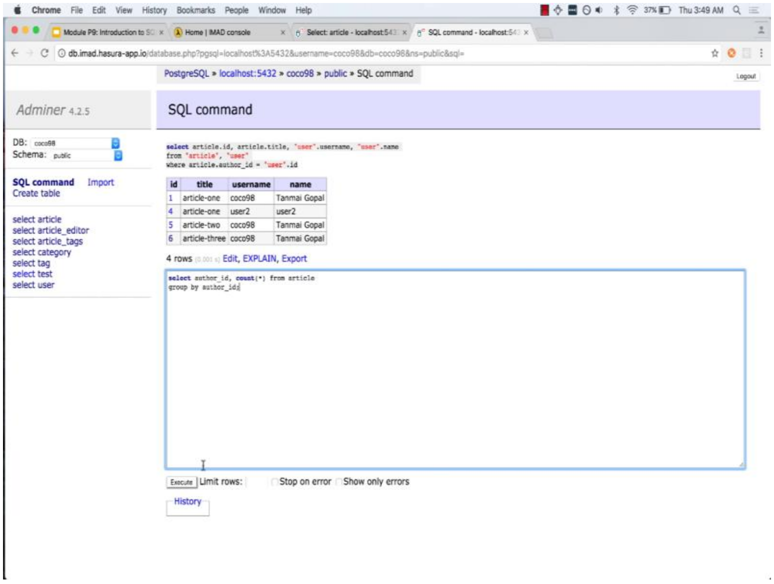
```
SELECT * FROM "article" LIMIT 50 (0.000 s) Edit
```

Modify	id	title	content ↓ =	author_id	category
edit	1	article-one	This is my first article	1	NULL
edit	4	article-one	this is my first article	2	NULL
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	work

(4 rows) whole result

Let us now look at another important kind of query. Let us go back to our article table let us clear these conditions, now the cross arrow. So, we have 4 articles. Now in these 4 articles, I want to make a query to fetch the number of the articles written by each author. So, I want to, so in this case the result that I want to be author id 1 has it one article has written three articles and author id two has written one article. So, the information that I want to fetch is how many article are written by each author.

(Refer Slide Time: 11:35)



SQL command

```
select article.id, article.title, "user".username, "user".name
from "article", "user"
where article.author_id = "user".id
```

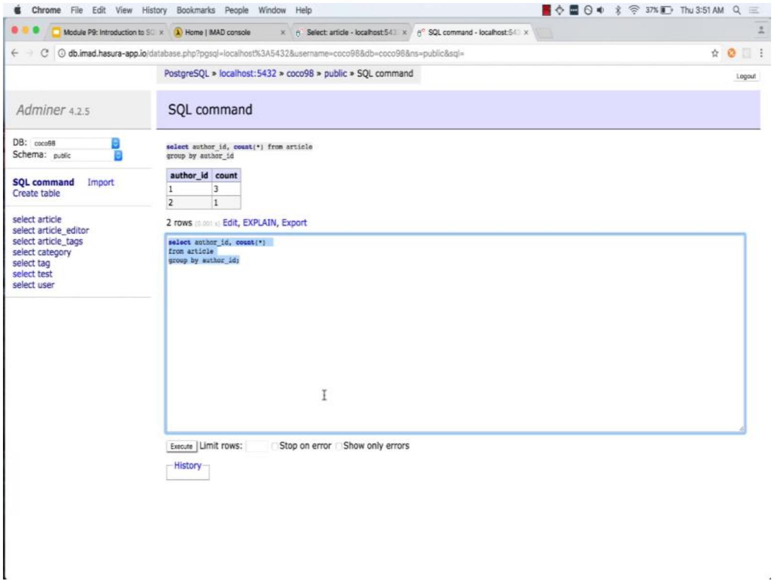
id	title	username	name
1	article-one	coco98	Tanmai Gopal
4	article-one	user2	user2
5	article-two	coco98	Tanmai Gopal
6	article-three	coco98	Tanmai Gopal

4 rows (0.001 s) Edit, EXPLAIN, Export

```
select author_id, count(*) from article
group by author_id
```

Let us look at what that SQL query looks like. So, I can say select author id and count of star from the article table and group by author id.

(Refer Slide Time: 11:56)



SQL command

```
select author_id, count(*) from article
group by author_id
```

author_id	count
1	3
2	1

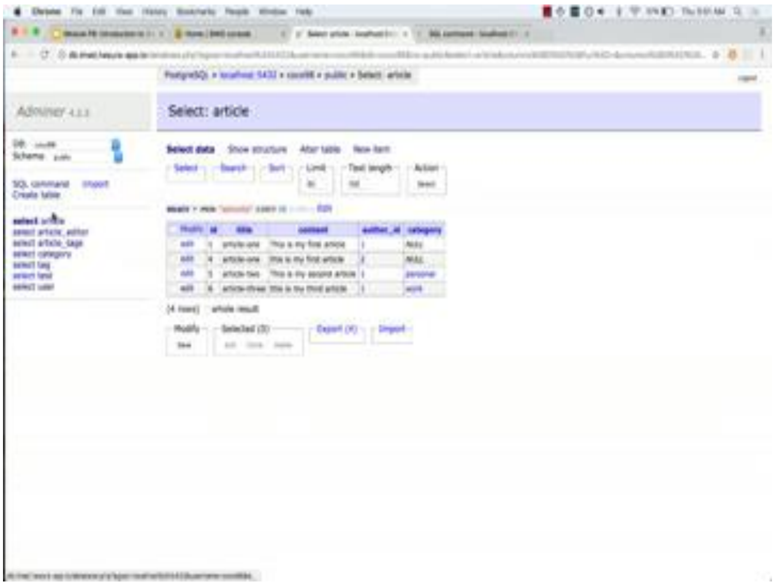
2 rows (0.001 s) Edit, EXPLAIN, Export

```
select author_id, count(*)
from article
group by author_id;
```

So, you can see that we have got the results. Let us read this SQL quickly, we are saying select the column author id and count of star and we will understand this is in bit. From the table article, but instead of just fetching all the data from the table article taking the data from the table article and group it by a particular columns value. So, group it by the values of author id.

And if you look it author id, they are only two values. So, it groups it for the value one and groups it for the value two; once you group a particular table, the only column that you can select is by performing what is called an aggregation on it. So, here we performed a count aggregation to get the number of articles that are written by this particular author. So, these queries are called aggregations or group by queries. And group by queries can be used to perform a lot of complex analytics on the data that is stored in your database.

(Refer Slide Time: 13:02)

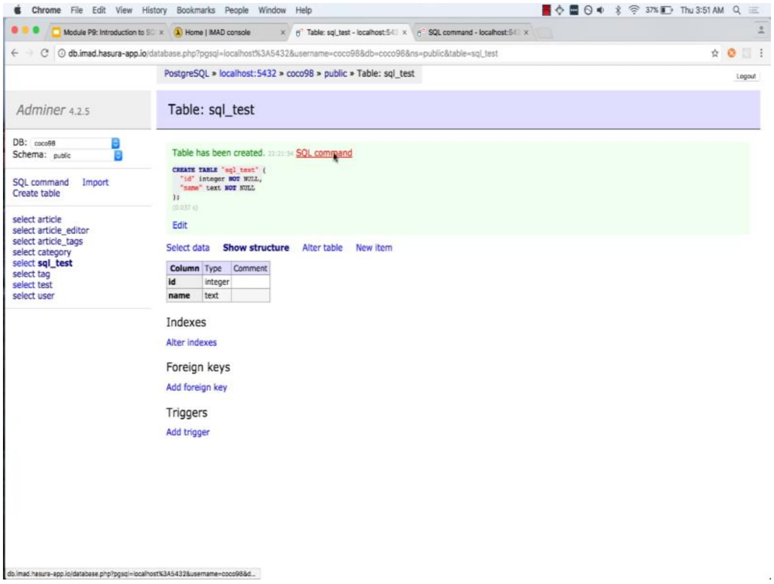


Modify	id	title	content	author_id	category
edit	1	article-one	This is my first article	1	[illegible]
edit	4	article-one	this is my first article	2	[illegible]
edit	5	article-two	This is my second article	1	personal
edit	6	article-three	this is my third article	1	work

(4 rows) whole result

Let us now look at something very different. Let us now actually create a table by using SQL. And what will do in this case is that we will actually just here Adminers, so we will go to create a table and we will say create a table called SQL test. Give it a column called id, now, which is in integer, and a column called name which is a text right and save this.

(Refer Slide Time: 13:21)



Chrome File Edit View History Bookmarks People Window Help
Thu 3:51 AM
Module P9: Introduction to SQ
Home | IMAD console
Table: sql_test - localhost:54
SQL command - localhost:54
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&table=sql_test
PostgreSQL » localhost:5432 » coco98 » public » Table: sql_test
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select article_editor
select article_tags
select category
select sql_test
select tag
select test
select user

Table: sql_test

Table has been created. 22:21:34 SQL command

```
CREATE TABLE "sql_test" (
  "id" integer NOT NULL,
  "name" text NOT NULL
);
```

(0.037 s)

Column	Type	Comment
id	integer	
name	text	

If you look at the create command, you will see what the SQL is and so let us copy paste this SQL. I am put into a command window right and let us change this to some other name SQL test 2.

And if we done this query, so let us redo these queries, these queries say - create a table and the table name with the following column. So, create table SQL test 2 with a column id and name; the id column is an integer column and null values are not allowed inside this column. The name column is a text column and no null values are allowed inside this column.

(Refer Slide Time: 13:58)

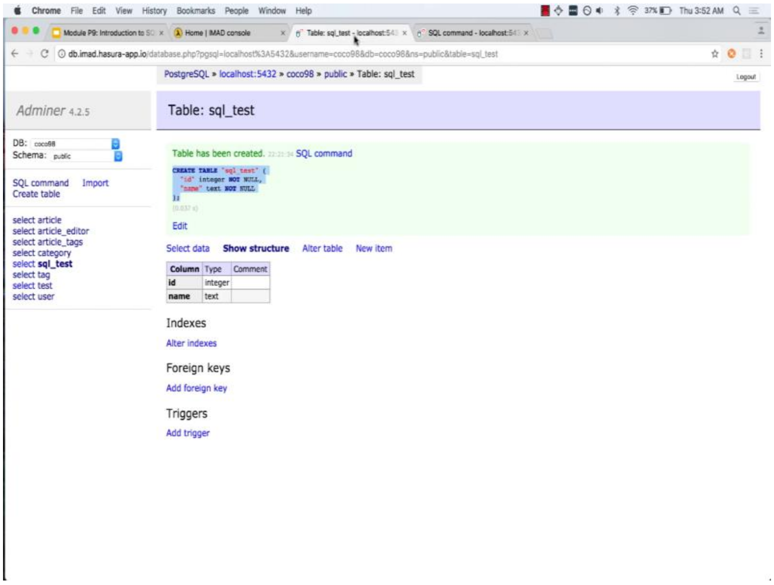

Chrome File Edit View History Bookmarks People Window Help
Thu 3:52 AM
Module P9: Introduction to SQ
Home | IMAD console
Table: sql_test - localhost:54
SQL command - localhost:54
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&table=sql_test
PostgreSQL » localhost:5432 » coco98 » public » Table: sql_test
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select article_editor
select article_tags
select category
select sql_test
select tag
select test
select user

Table: sql_test

Table has been created. 22:21:34 SQL command

```
CREATE TABLE "sql_test" (
  "id" integer NOT NULL,
  "name" text NOT NULL
);
```

(0.037 s)

Edit

Select data **Show structure** Alter table New item

Column	Type	Comment
id	integer	
name	text	

Indexes

Alter indexes

Foreign keys

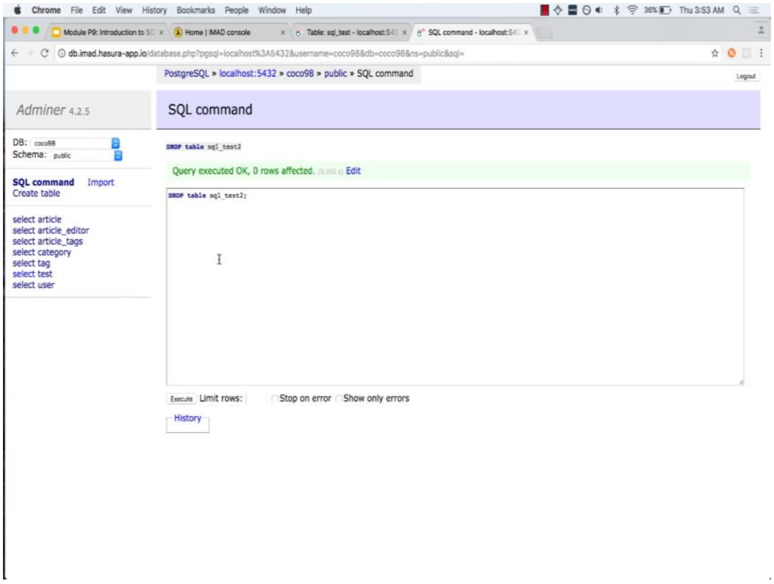
Add foreign key

Triggers

Add trigger

If you click on execute, you will see that another table has been created. Similarly we can drop these tables or rather we can delete these tables and the way to delete these tables is called a drop.

(Refer Slide Time: 14:07)


Chrome File Edit View History Bookmarks People Window Help
Thu 3:53 AM
Module P9: Introduction to SQ
Home | IMAD console
Table: sql_test - localhost:54
SQL command - localhost:54
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&sql=
PostgreSQL » localhost:5432 » coco98 » public » SQL command
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command
Import
Create table
select article
select article_editor
select article_tags
select category
select tag
select test
select user

SQL command

```
DROP table sql_test2
```

Query executed OK, 0 rows affected. (0.352 s) Edit

```
DROP table sql_test2;
```

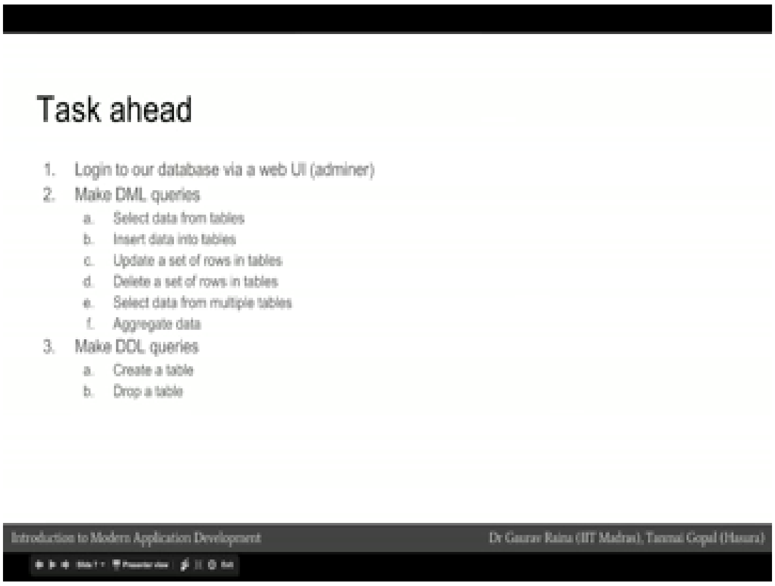
Execute Limit rows: □ Stop on error □ Show only errors

History

Let us do a drop table SQL underscore test. You can see that the table has disappeared.

Let us draw the SQL test 2 and it struck. So, this SQL statement deletes tables this is called dropping the table. So, in SQL you would typically refer to dropping deleting a table as a drop, you delete a data base you can also delete an entire data base by doing a drop data base, but when you in, but when you a delete a row the SQL command is called a delete not a drop.

(Refer Slide Time: 14:50)



Task ahead

1. Login to our database via a web UI (adminer)
2. Make DML queries
   a. Select data from tables
   b. Insert data into tables
   c. Update a set of rows in tables
   d. Delete a set of rows in tables
   e. Select data from multiple tables
   f. Aggregate data
3. Make DDL queries
   a. Create a table
   b. Drop a table

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

This is just a super fast introduction to what SQL is and what the power of SQL is. I did not go into any syntax to explain to you on what the SQL language is and what the syntax of language is, but the idea was just to go there few fundamentals of what the power of SQL is.

(Refer Slide Time: 15:09)

Learn SQL!

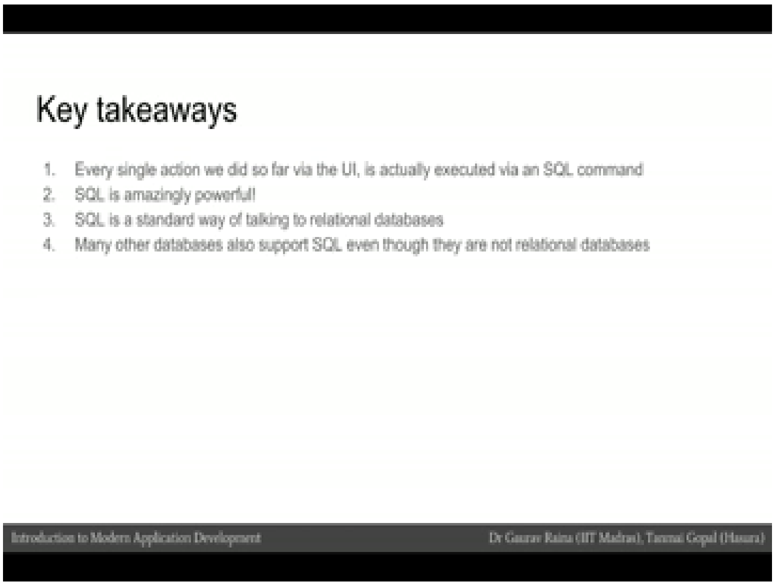
https://www.codecademy.com/learn/learn-sql

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

I would encourage those of you interested in the back end development side of things, in interested in databases to definitely learn SQL. SQL is almost one of the prerequisites for doing complex analytics and for analyzing databases and managing databases.

(Refer Slide Time: 15:26)



Key takeaways

1. Every single action we did so far via the UI, is actually executed via an SQL command
2. SQL is amazingly powerful!
3. SQL is a standard way of talking to relational databases
4. Many other databases also support SQL even though they are not relational databases

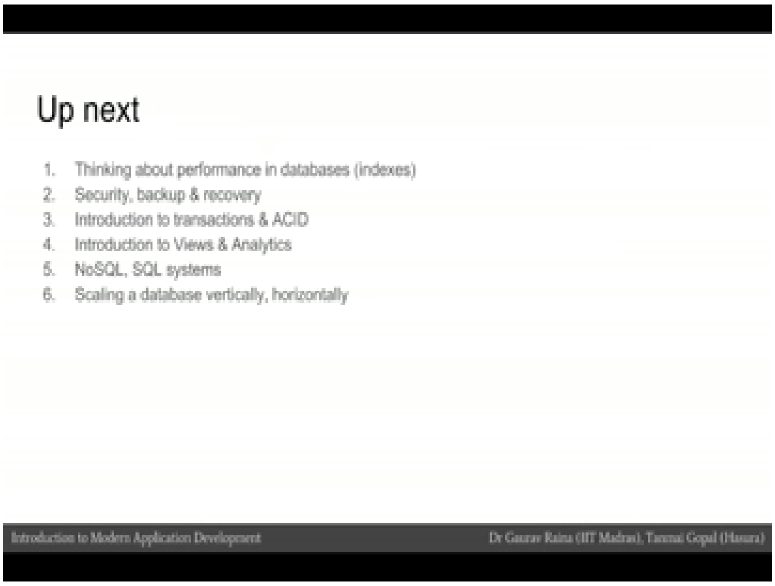
Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

The important things you will remember after this from this module are number one what the SQL syntax looks like, and that the SQL language is extremely powerful and every single thing that we do via the UI on Adminer is actually happening via an UI SQL command. The index is that you make, indexes that you delete, the primary keys that you make, when you change a type of a column, when you insert data or search for data in a particular way all of those are SQL commands.

SQL is standard way of talking to relational databases and in fact, Postgres my SQL, oracle, my MySQL server, psi base all of them are respect to the SQL standard. And each of these databases differ a little bit in the particular version of SQL or the particular syntax of SQL they use, but the core language is the same across all these databases. A lot of other data bases which are not relational databases also support SQL.

(Refer Slide Time: 16:23)



Up next

1. Thinking about performance in databases (indexes)
2. Security, backup & recovery
3. Introduction to transactions & ACID
4. Introduction to Views & Analytics
5. NoSQL, SQL systems
6. Scaling a database vertically, horizontally

Introduction to Modern Application Development

Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

Up next are several quick theory modules on thinking about performance in databases especially touching on indexes, security, backup, recovery, and introduction to transactions in ACID, which are very important for financial applications or e commerce applications, and introduction to views, analytics. We will also look at the difference between NoSQL and SQL systems. And then finally, look at how we can scale a database or how databases are scaled vertically and horizontally.