In this lesson we are going to talk about the SERIAL type. We have learned how to create auto-incrementing fields and sequences. That was a bit complicated and it required a few steps. Fortunately, now that we know how that works, we can tall you that Postgres has a shortcut, so we do not need to create a sequence and give it the owner and set it to start at a certain number and things like that. Instead when we create our table, we can just type in an id of type serial and that creates the sequence for us and all that stuffs.

SELECT TABLE test (

id SERIAL PRIMARY KEY,

name text

);

The id that we declared would create an auto-incrementing id, starting from 1 and then there is a name column of the type text. Text is just like character varying but it does not have an upper limit as a size. If we want to limit the size, we can declare the character varying type with the upper limit declared in parentheses. If we do not want to limit the size, we can declare text as our datatype.

The id has the SERIAL type with the PRIMARY KEY constraint. After running the query, it runs successfully, and out table gets set up. Now we will INSERT data into the table, in our name field.

INSERT INTO test(name) VALUES (‘Jose’);

After running the query, it runs successfully, thus the name gets entered in out table. Now we will view the table,

SELECT \* FROM test;

|  |  |
| --- | --- |
| id  integer | name  text |
| 1 | Jose |

This is how we can simply the sequence and auto-incrementing ids. But we did not want to introduce it initially, because the sequence is how the SERIAL type is built in the background, so it is important us to know that the sequence is there in the background and therefore that is how the id is generated and the sequence is only going to generate numbers one after another and that is it. The sequence itself does not guarantee unique numbers being generated, just guarantees that there is one after another. If we want to make sure that the id is always unique, then just give it the PRIMARY key constraint or we can give it the UNIQUE constraint. In the case of this table we have given it the PRIMARY key constraint. So, if the SERIAL field generates a number that is already there, then it will just go on to the next one, so it is quite nice as well.

That is how we use a SERIAL datatype.