**PostgreSQL**

* It is widely used by top companies like apple, witch or Instagram or even Nasa uses PostgreSQL to manage their data.
* It is the World’s Most Advanced Open-Source Relational Database.
* It is Free and Open Source
* It is a great Career Opportunities after you learn PostgreSQL
* It has great Community Support

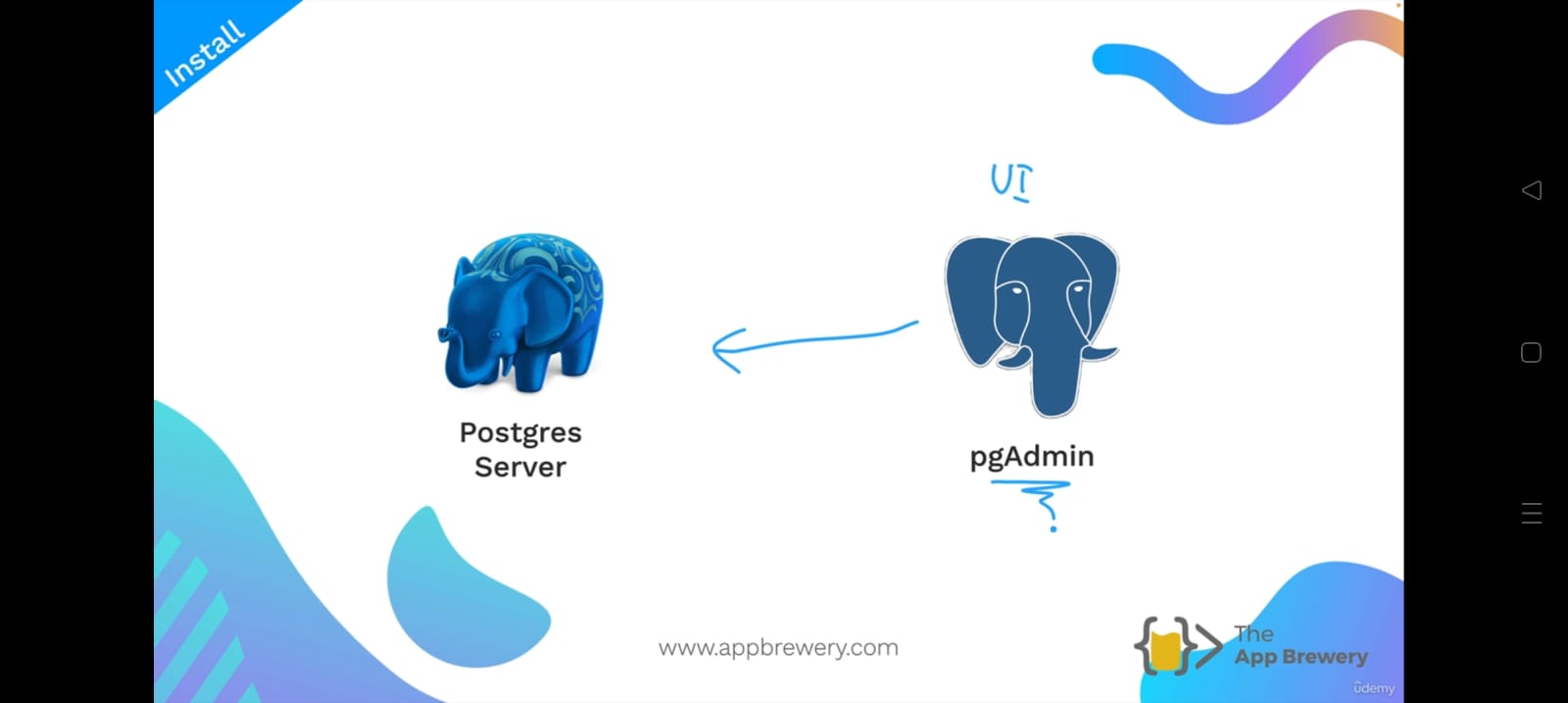
Client

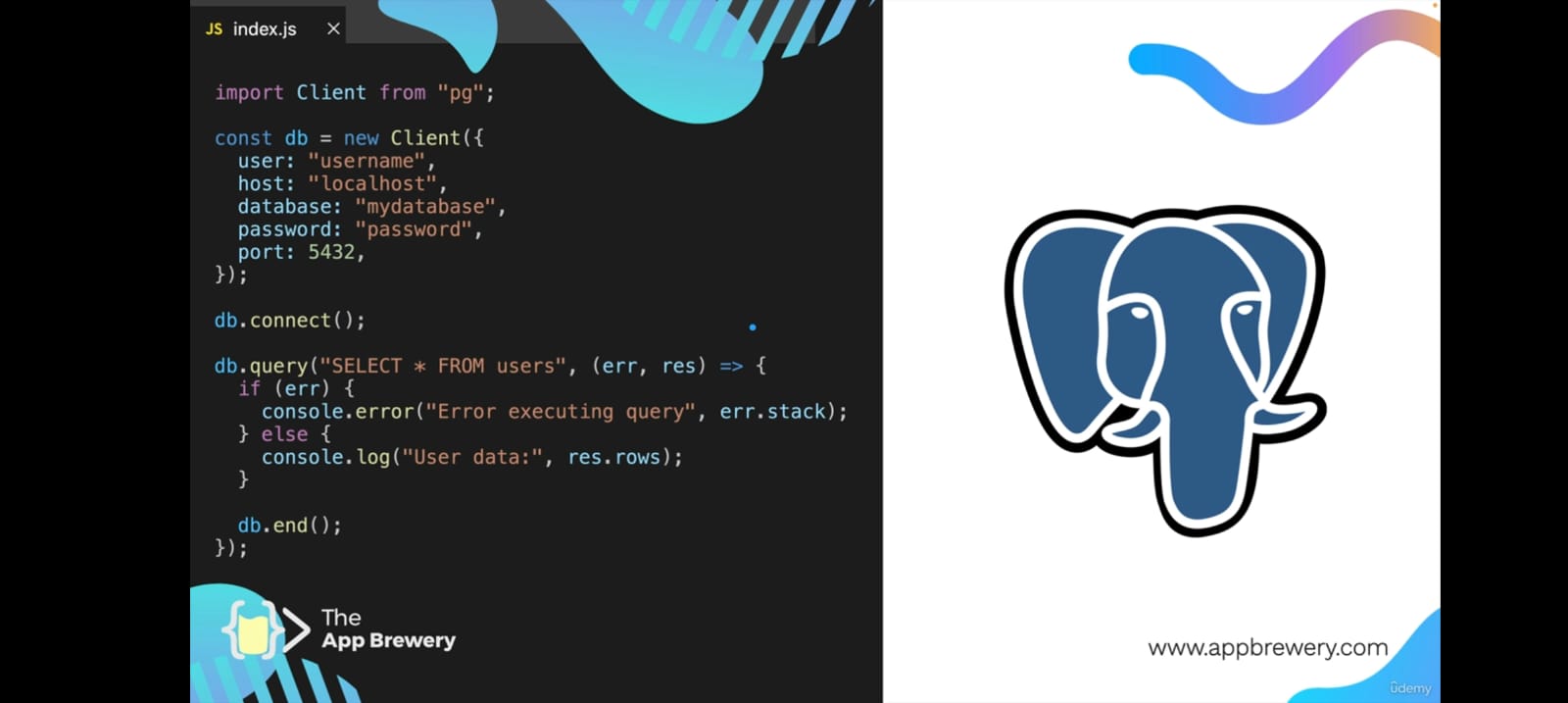
Server

Logic

DB

We require 2 pieces of this database:



Example Code:

**How to Install PostgreSQL and pgAdmin**

Installation Instructions:

**1. Windows Users:** Download the Postgres Installer here:

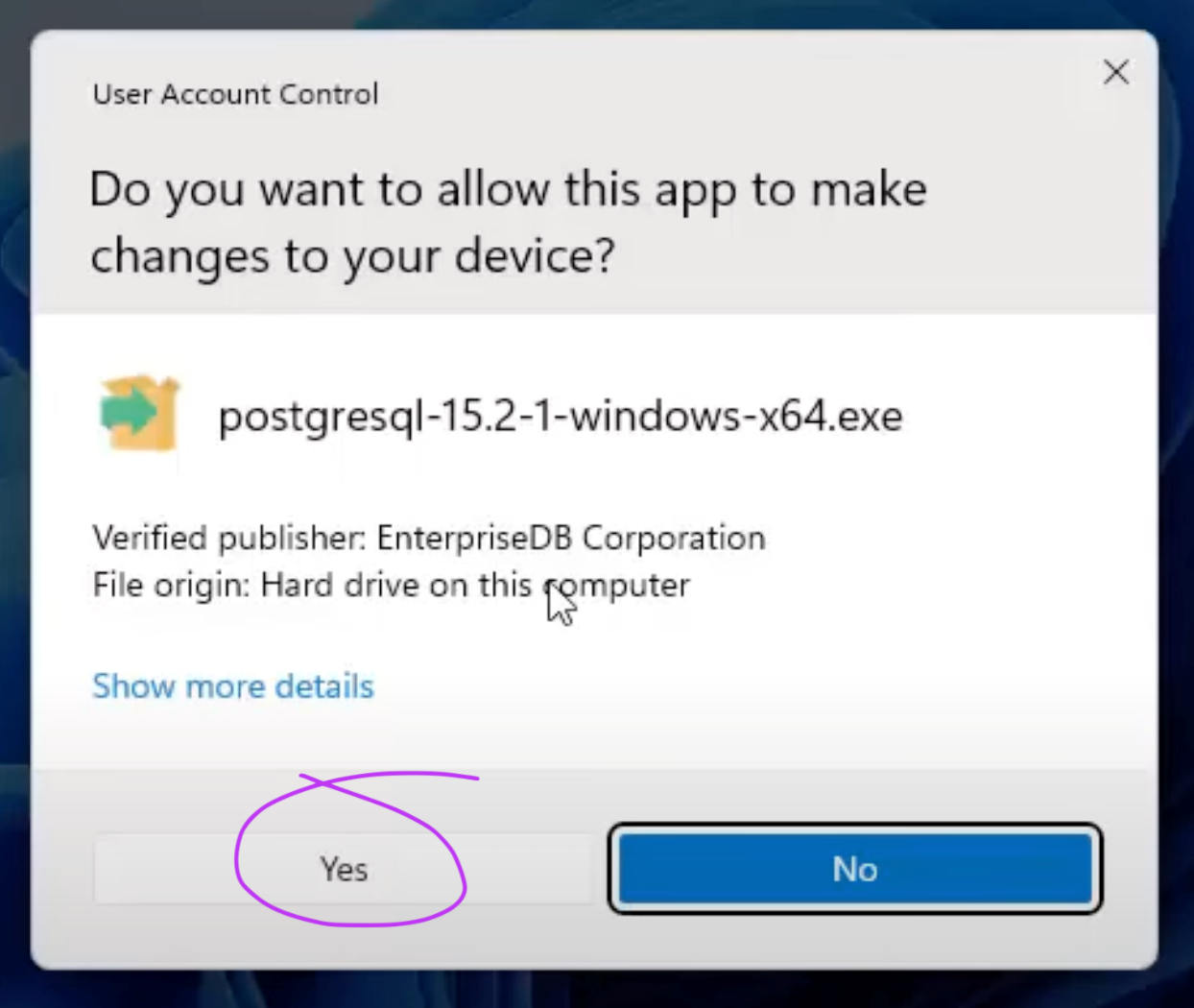
<https://sbp.enterprisedb.com/getfile.jsp?fileid=1258649>

**1. Mac Users:** Download the Postgres Installer here:

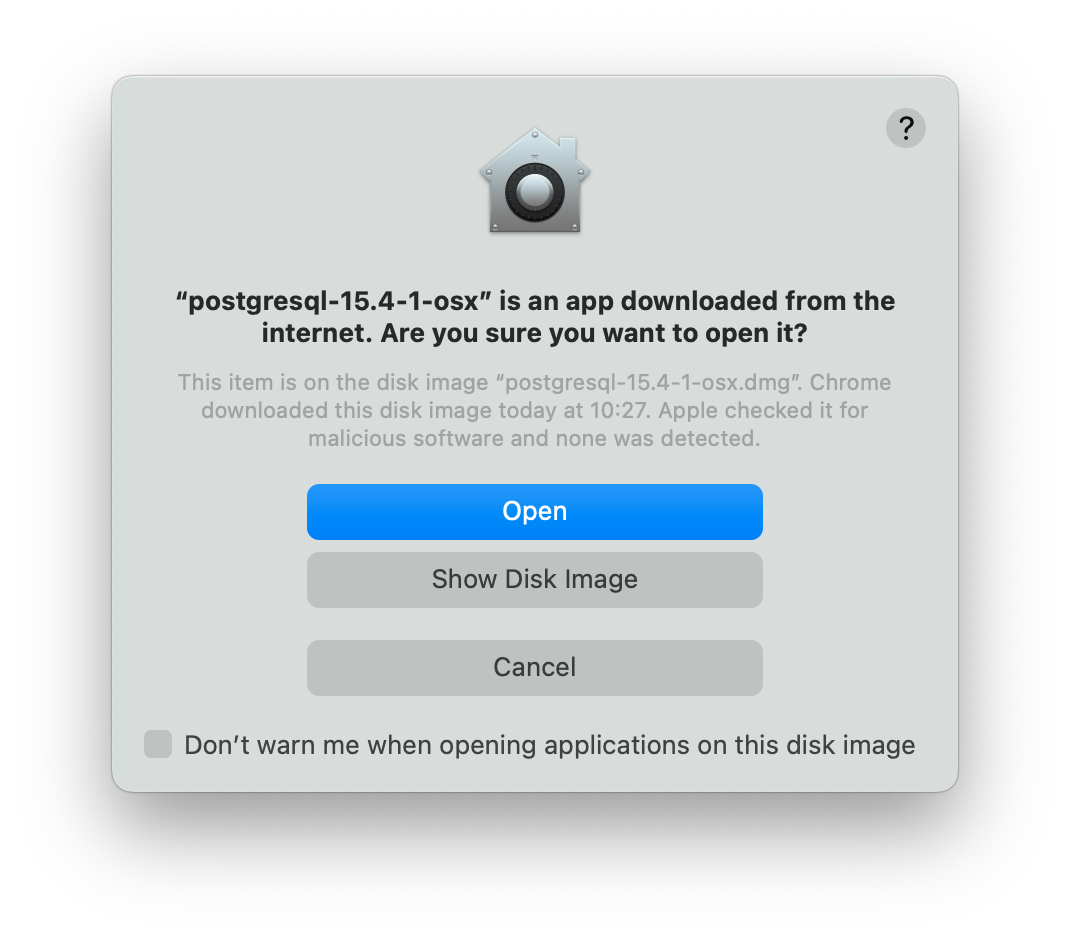
<https://sbp.enterprisedb.com/getfile.jsp?fileid=1258653>

2. Double click on the downloaded file to start the installer.

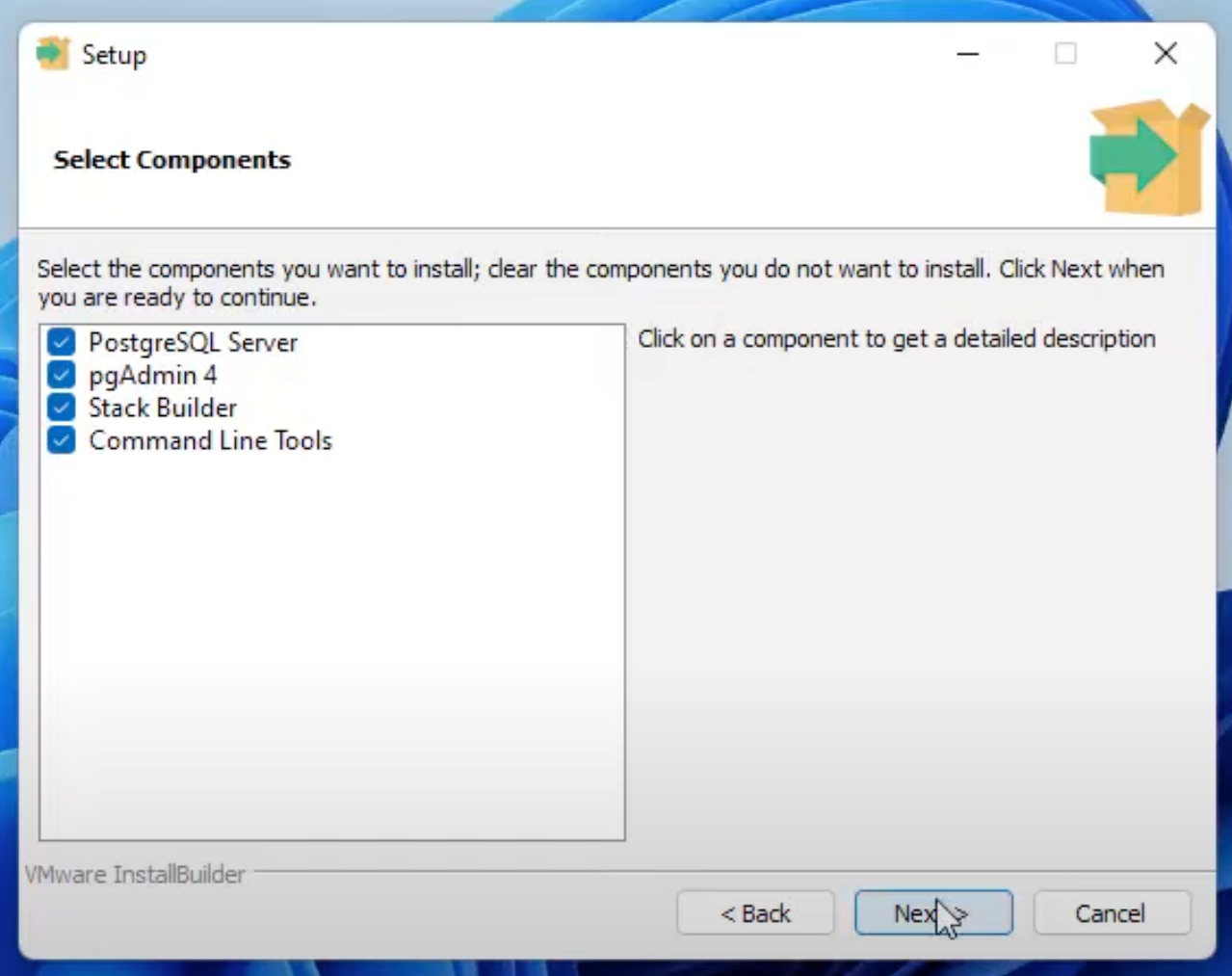
**3. Windows Users:**If you see a pop up asking for permission to run the installer, select YES.



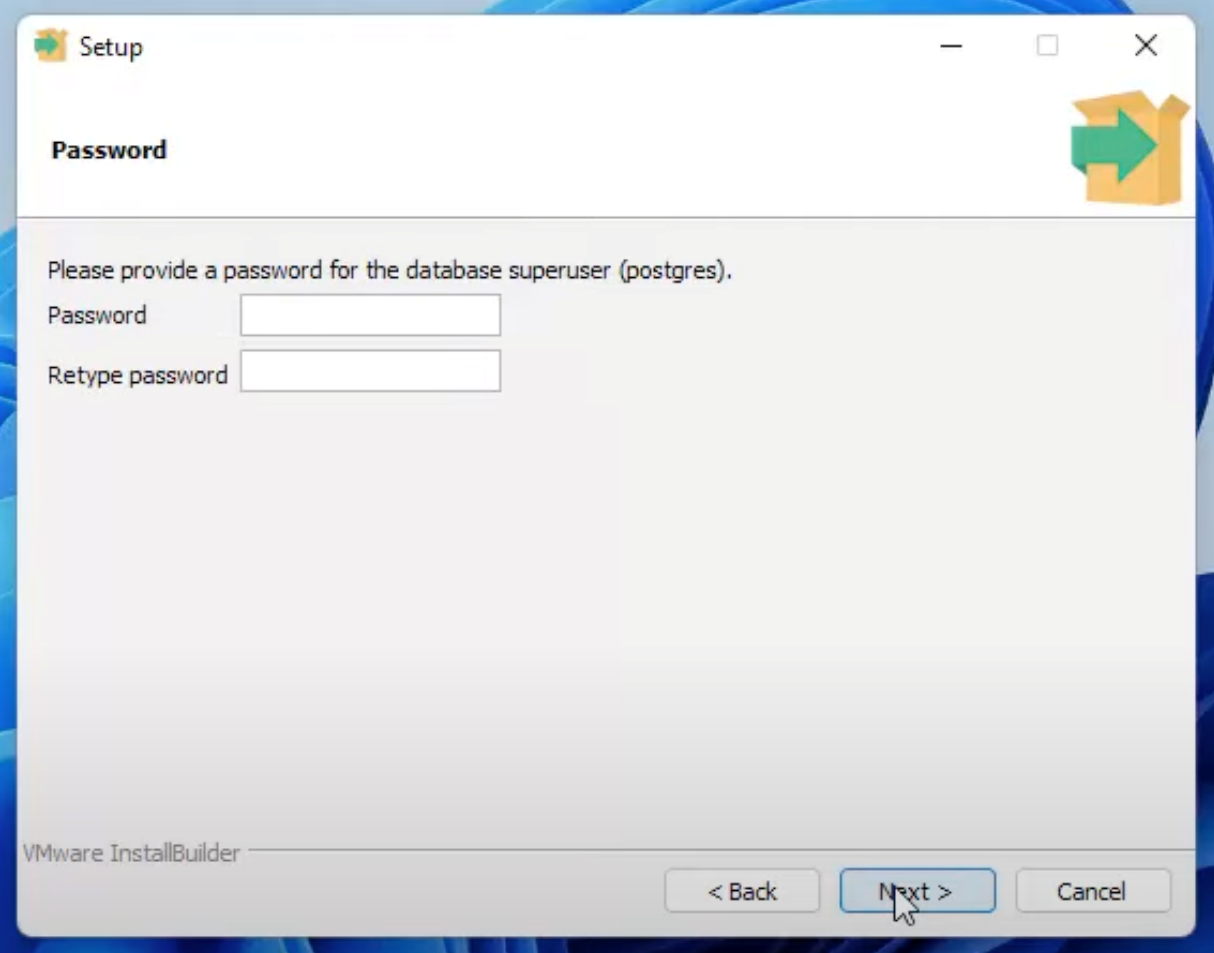
**3. Mac Users:**If you see a pop up asking for permission to run the installer, select Open, you might need to enter your Mac password to allow the installer to run.



4. Click **Next** to continue through the installer, until you reach this pane and make sure that everything is selected especially **pgAdmin**.



5. Continue clicking **Next** until you reach this screen. Your superuser username is **postgres** and you need to set a password. **Make sure you write this password down**. We will need this later to access our database.



6. Continue through the installer, leave everything as the default and just keep clicking **Next**. Once installation completes then you're done you can click **Finish** and you can continue to the next lesson!

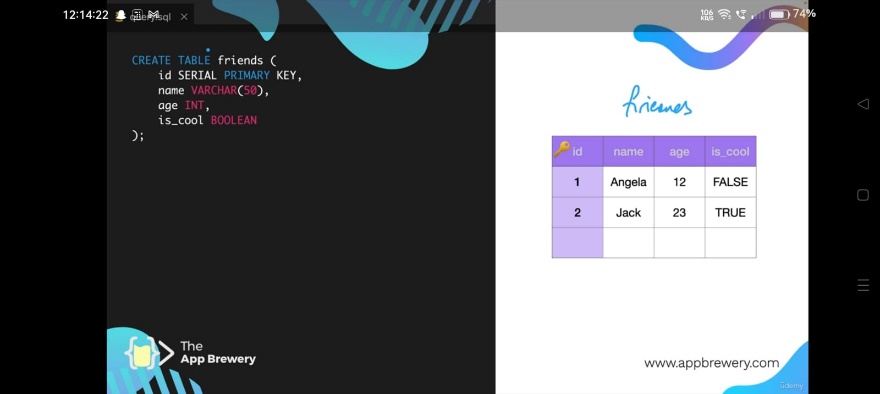
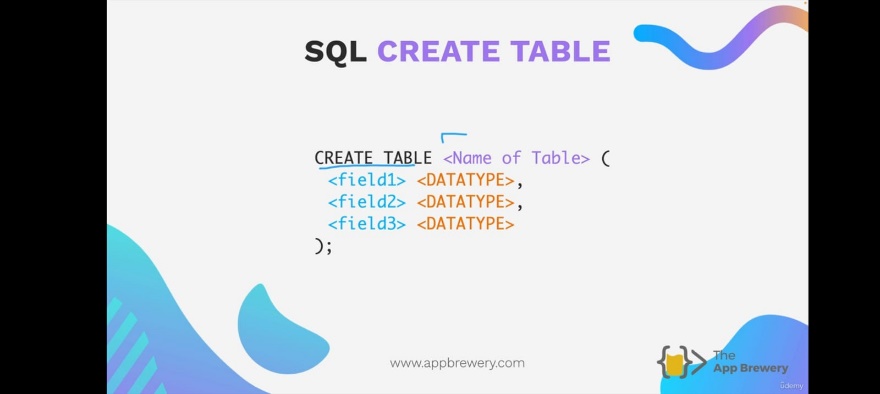
NOTE: If Stack Builder launches after installation completes, just close it. We will be doing everything from scratch together in the next lesson without using a wizard.

My user details:

Username: postgres (default)

Password: 12345

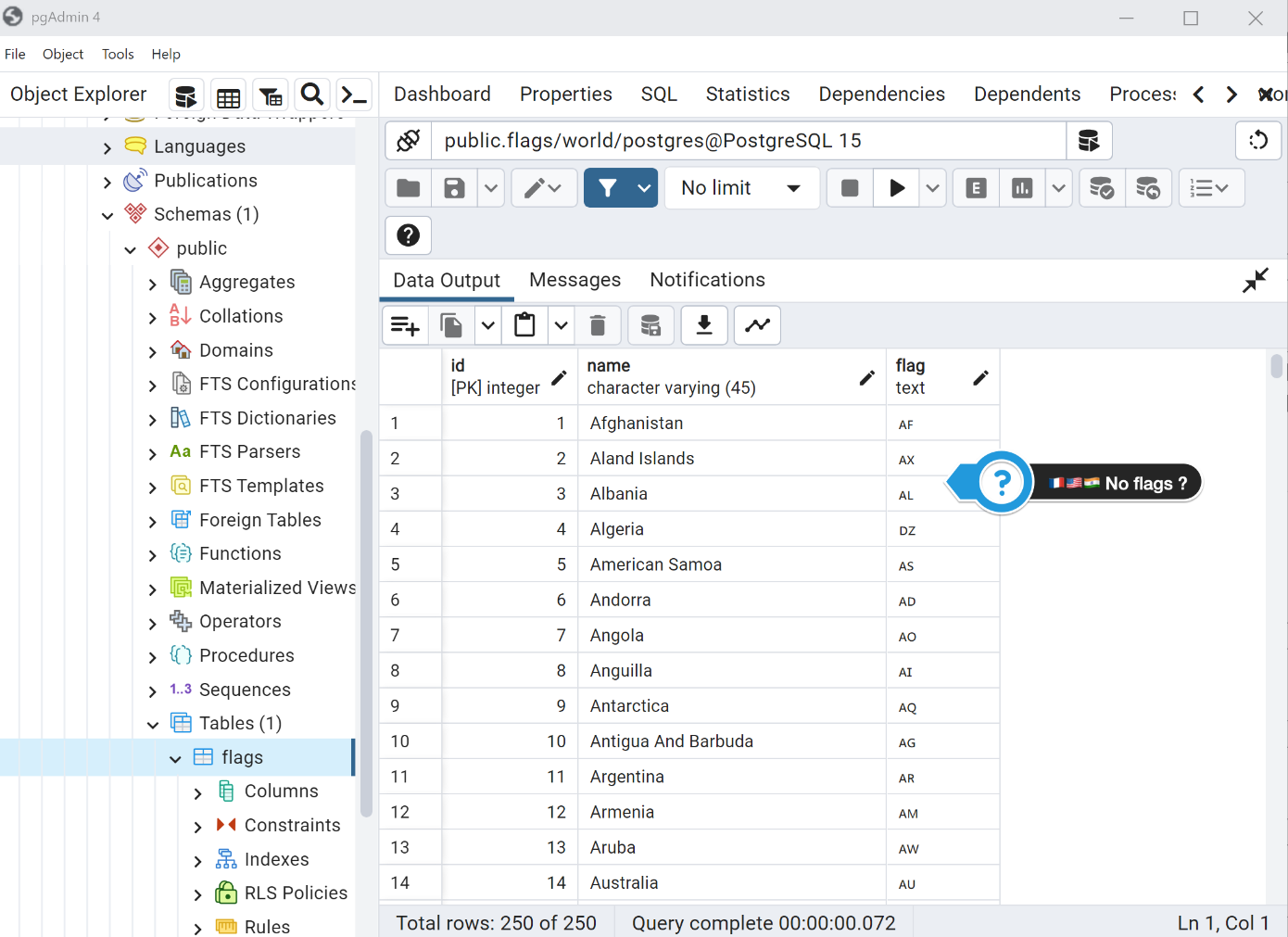
SQL Create Table syntax and example:



**Flag Emojis on Windows ?**

Windows does not display flag images for emoji

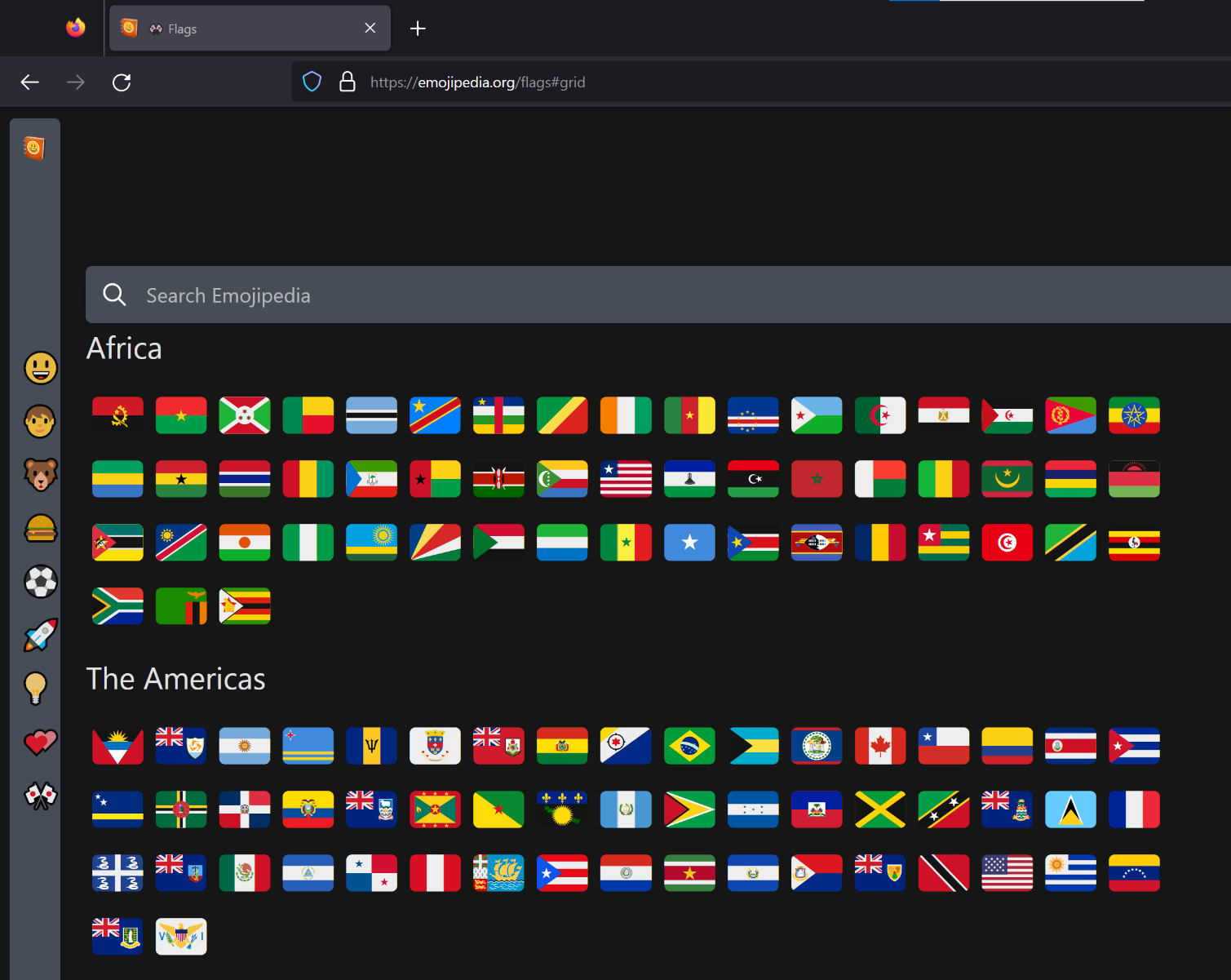
In the next couple of lessons you will see various country's flags being displayed in the videos. However, Microsoft Windows does not natively display the images of flags and you won't see flags in VS Code or pgAdmin. Instead, Windows will show you the country code which looks this:



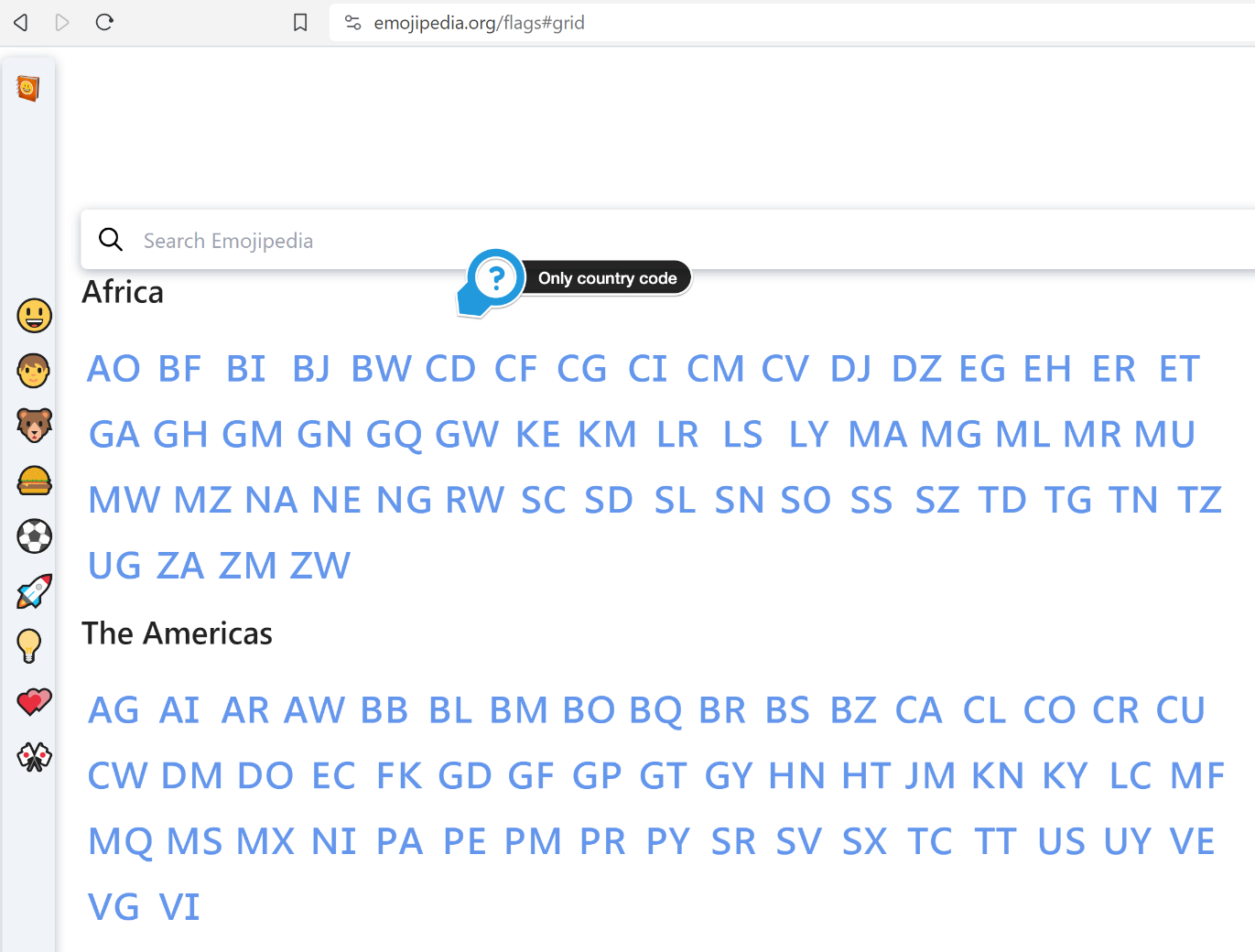
**But don't worry!** The important thing for our web development projects is how the flags are displayed in the browser (and *not* how the emojis are displayed in .csv files or pgAdmin).

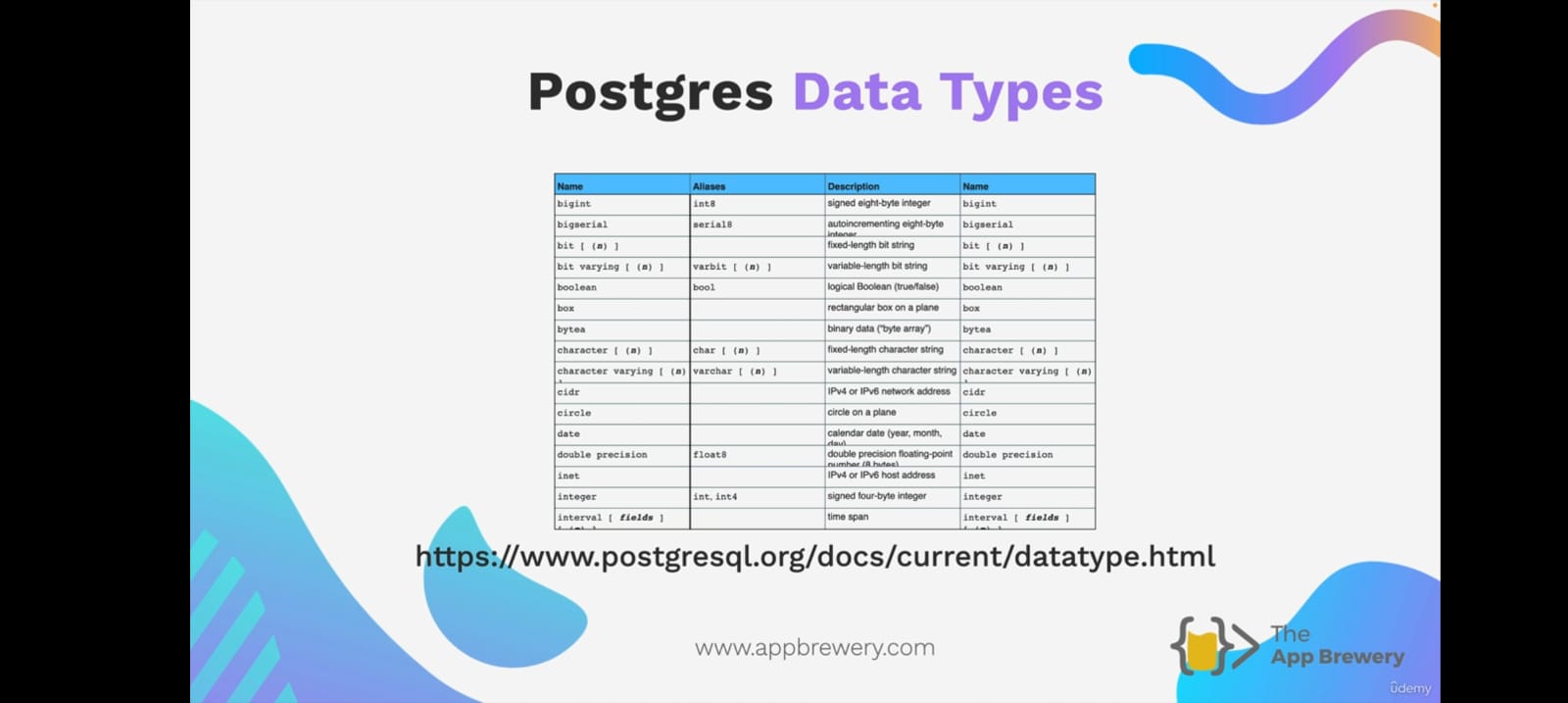
Use Mozilla Firefox for the Flag Quiz Project

As a Windows user, the easiest way to complete upcoming country flag quiz project is to simply use the Mozilla [Firefox](https://www.mozilla.org/en-GB/firefox/new/) browser (instead of Chrome). Firefox displays flags just fine:

[](https://emojipedia.org/flags)

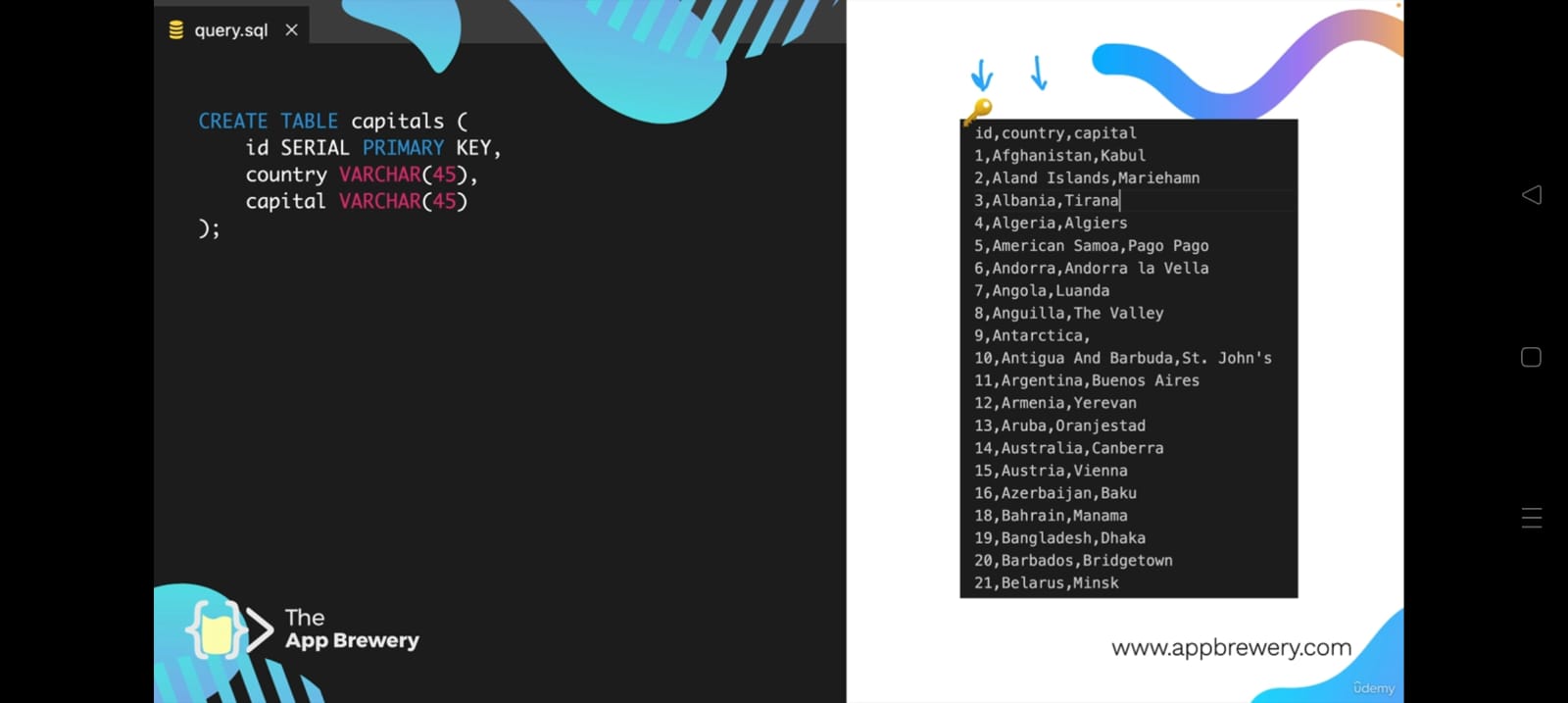
Here's Google Chrome for comparison:



 Postgres Data Types:

csv – comma separated values

If we copy excels data by rows and columns and paste it into notepad then it will give data in the format of “comma separated values” and vise versa.

Example:

First program on vs code to get all data:

import pg from "pg";

*const* db = new pg.Client({

    user:"postgres",

    password:"12345",

    database:"userDB",

    host:"localhost",

    port:5432

})

db.connect()

db.query("SELECT \* FROM capitals",(*err*,*res*)*=>*{

    if(*err*) console.log(*err*.stack);

    else console.log("User data is ",*res*.rows)

    db.end()

})

Implementing a simple CHART JS and passing data from PostgreSQL to nodeJS to EJS in script

Index.js:

import express  from "express";

import pg from "pg";

*const* db = new pg.Client({

    port:5432,

    host:"localhost",

    database:"userDB",

    user:"postgres",

    password:"12345"

})

db.connect()

*const* app = express()

*const* PORT = 3000

app.set("view engine","ejs")

app.use(express.static("public"))

app.get("/",async (*req*,*res*)*=>*{

    try{

*const* result = await db.query("SELECT \* FROM world\_food")

*let* countryCategories = [];

*let* rice\_population = []

*let* wheat\_population = []

    for(*let* row of result.rows){ countryCategories.push(row.country); rice\_population.push(row.rice);wheat\_population.push(row.wheat) }

*res*.render("world\_food",{countryCategories,rice\_population,wheat\_population})

}catch(err){

*res*.send(err)

}

})

app.listen(PORT,()*=>*{

    console.log(`Server is running at port ${PORT}`)

})

world\_foold.ejs:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>World Food Data</title>

    <style>

        \*{

*margin*: 0;

*padding*: 0;

        }

        #chart{

*position*: absolute;

*top*: 50%;

*left*: 50%;

*transform*: translate(-50%,-50%);

*width*: 60rem;

*box-shadow*: 0 0 5px black;

*border-radius*: 2rem;

*padding*: 1rem;

        }

    </style>

</head>

<body>

    <div id="chart"></div>

    <script src="https://cdn.jsdelivr.net/npm/apexcharts"></script>

    <script>

*var* options = {

          series: [{

          name: 'Rice',

          data: "<%= rice\_population %>".split(",")

        }, {

          name: 'Wheat',

          data: "<%= wheat\_population %>".split(",")

        }],

          chart: {

          type: 'bar',

          height: 350

        },

        plotOptions: {

          bar: {

            horizontal: false,

            columnWidth: '55%',

            endingShape: 'rounded'

          },

        },

        dataLabels: {

          enabled: false

        },

        stroke: {

          show: true,

          width: 2,

          colors: ['transparent']

        },

        xaxis: {

          categories: "<%= countryCategories %>".split(","),

        },

        yaxis: {

          title: {

            text: 'Population'

          }

        },

        fill: {

          opacity: 1

        },

        tooltip: {

          y: {

            formatter: *function* (*val*) {

              return " Population:"+*val*

            }

          }

        }

        };

*var* chart = new ApexCharts(document.querySelector("#chart"), options);

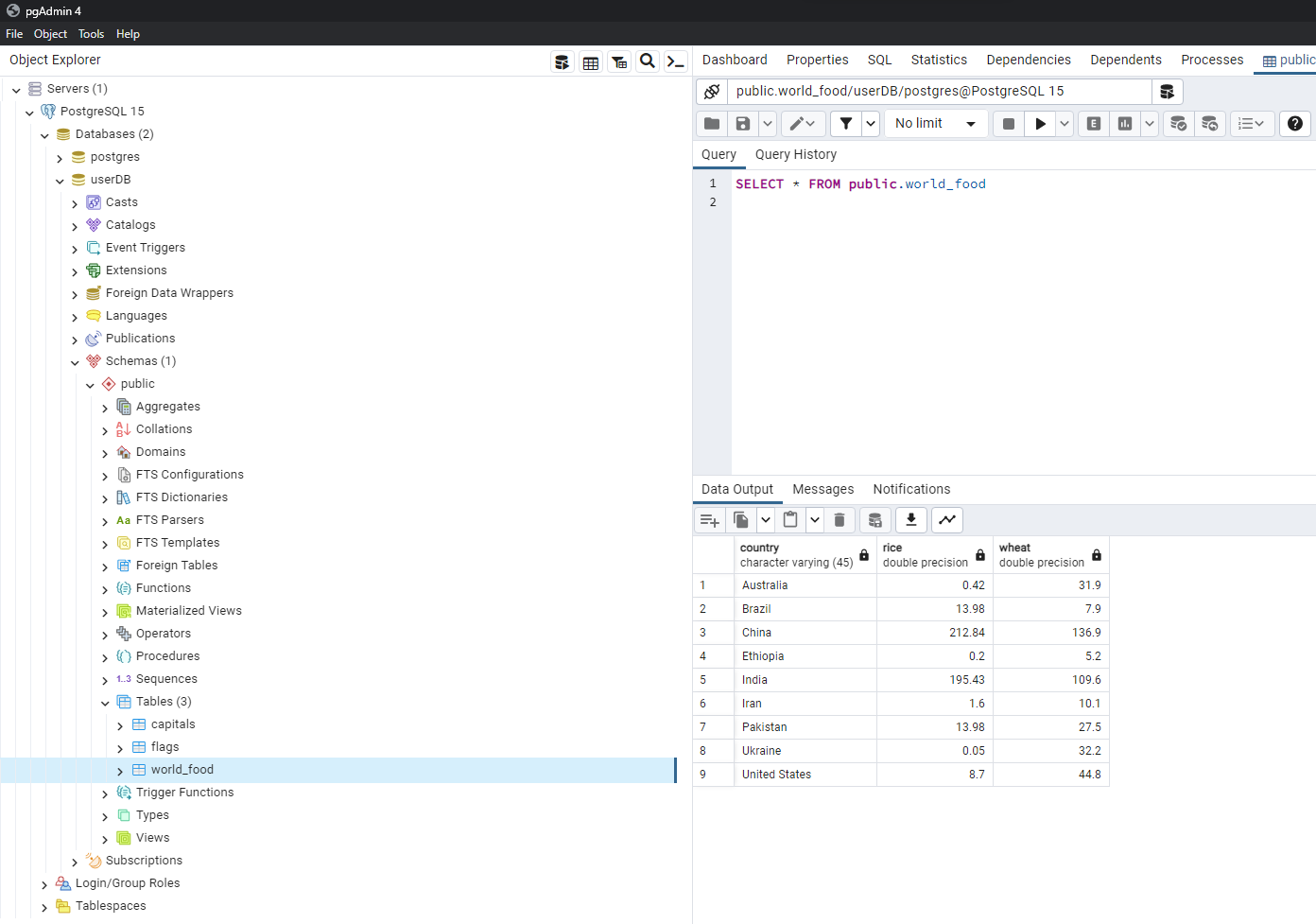
        chart.render();

    </script>

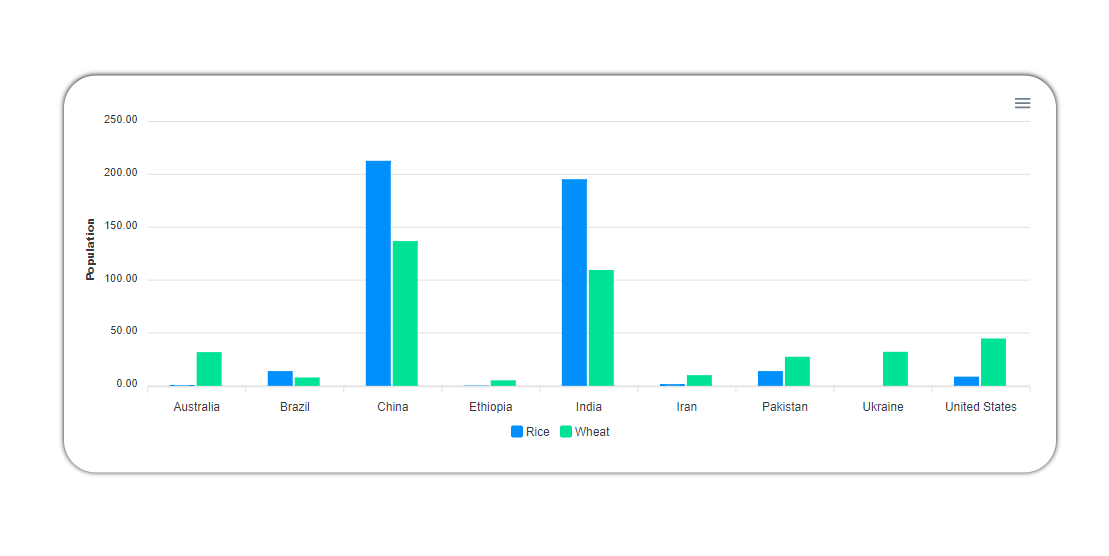
</body>

</html>

PostgreSQL data:



Output:



Important that should know it needs:

*const* expSynt1 = await db.query("SELECT \* FROM world\_food WHERE Country LIKE 'U' || '%' || 'e'")

*const* expSynt2 = await db.query("SELECT \* FROM world\_food WHERE Country = $1",["United States"])