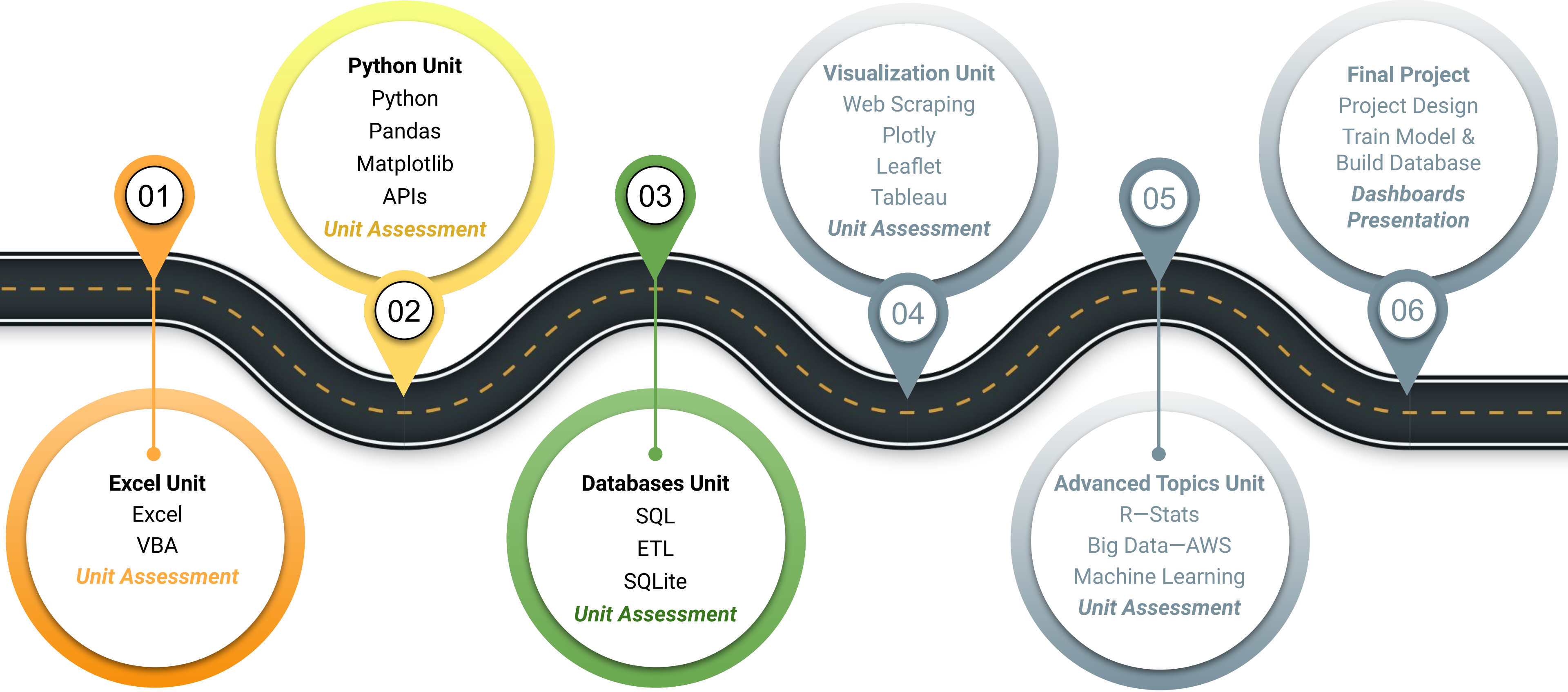
SQL

Data Boot Camp

Lesson 7.1

The Big Picture



01

Excel Unit

Excel

VBA

Unit Assessment

02

Python Unit

Python

Pandas

Matplotlib

APIs

Unit Assessment

03

Databases Unit

SQL

ETL

SQLite

Unit Assessment

04

Visualization Unit

Web Scraping

Plotly

Leaflet

Tableau

Unit Assessment

05

Advanced Topics Unit

R—Stats

Big Data—AWS

Machine Learning

Unit Assessment

06

Final Project

Project Design

Train Model &
Build Database

Dashboards
Presentation

Pro Tip:

As we dive into our first database, remember that it might be a slight shift in your method of thinking. But don’t worry! It’ll become second nature soon.

Module 7

This Week: SQL

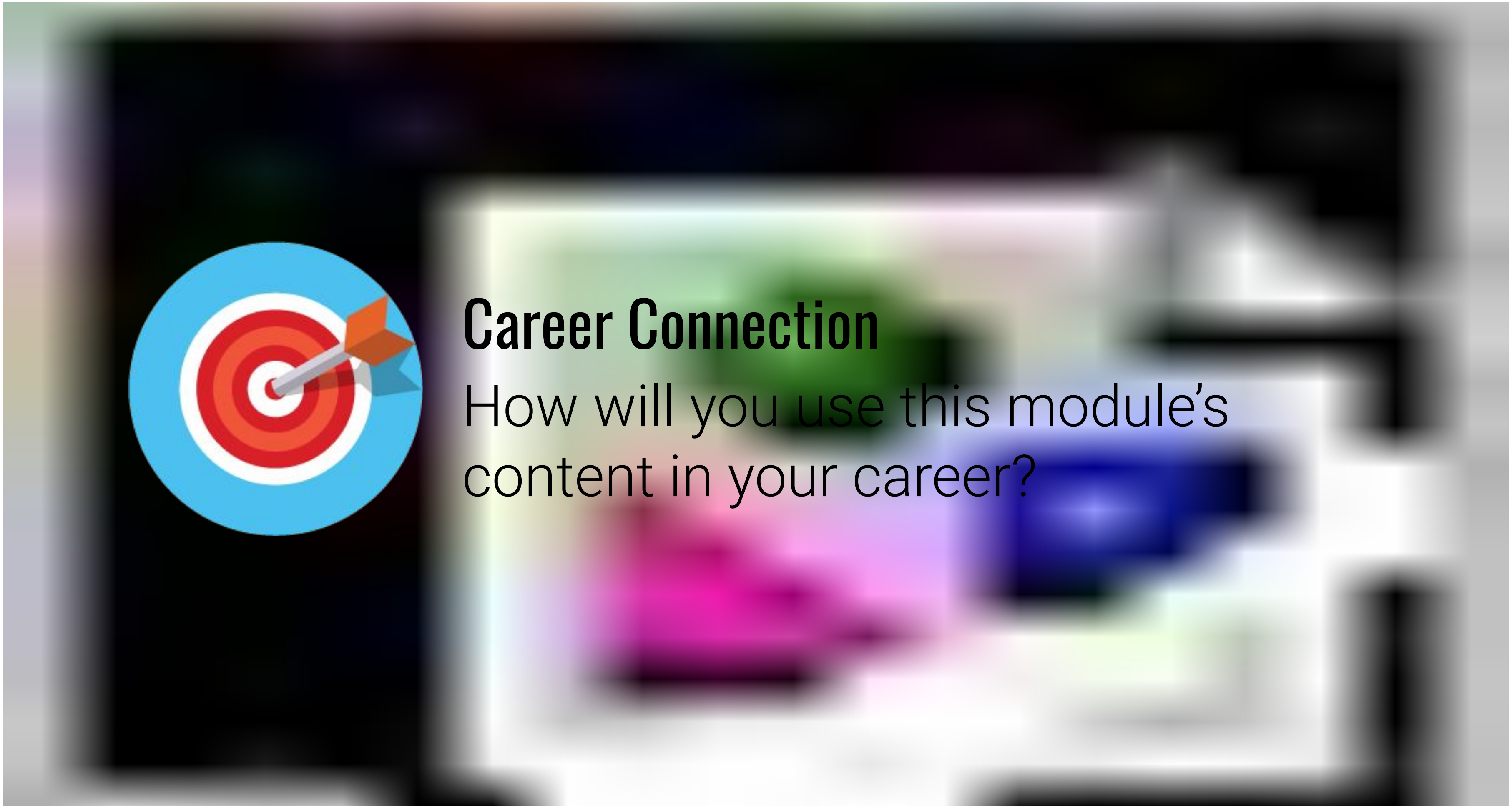
This Week: SQL

By the end of this week, you'll know how to:

- Design an ERD that will apply to the data.
- Create and use a SQL database.
- Import and export large CSV datasets into pgAdmin.
- Practice using different joins to create new tables in pgAdmin.
- Write basic- to intermediate-level SQL statements.

This Week's Challenge

Using the skills learned throughout the week, create two tables that would help a company determine employee eligibility for a mentorship program.



Career Connection

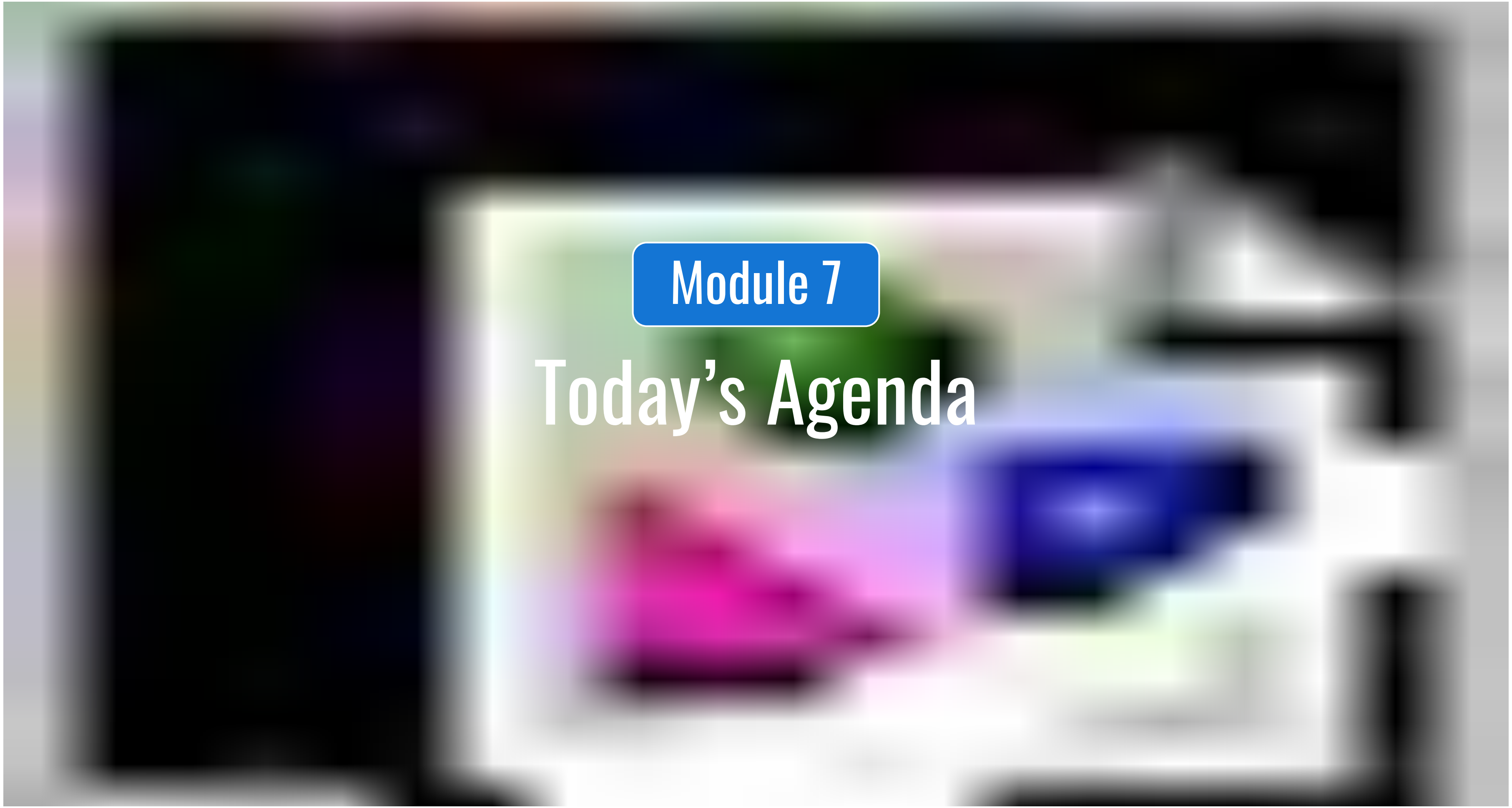
How will you use this module’s content in your career?

Module 7

How to Succeed This Week

Pro Tip:

Take full advantage of office hours and your support network as we shift over to working with databases!



Module 7

Today’s Agenda

Today’s Agenda

By completing today’s activities, you’ll learn the following skills:

01 CRUD Operations

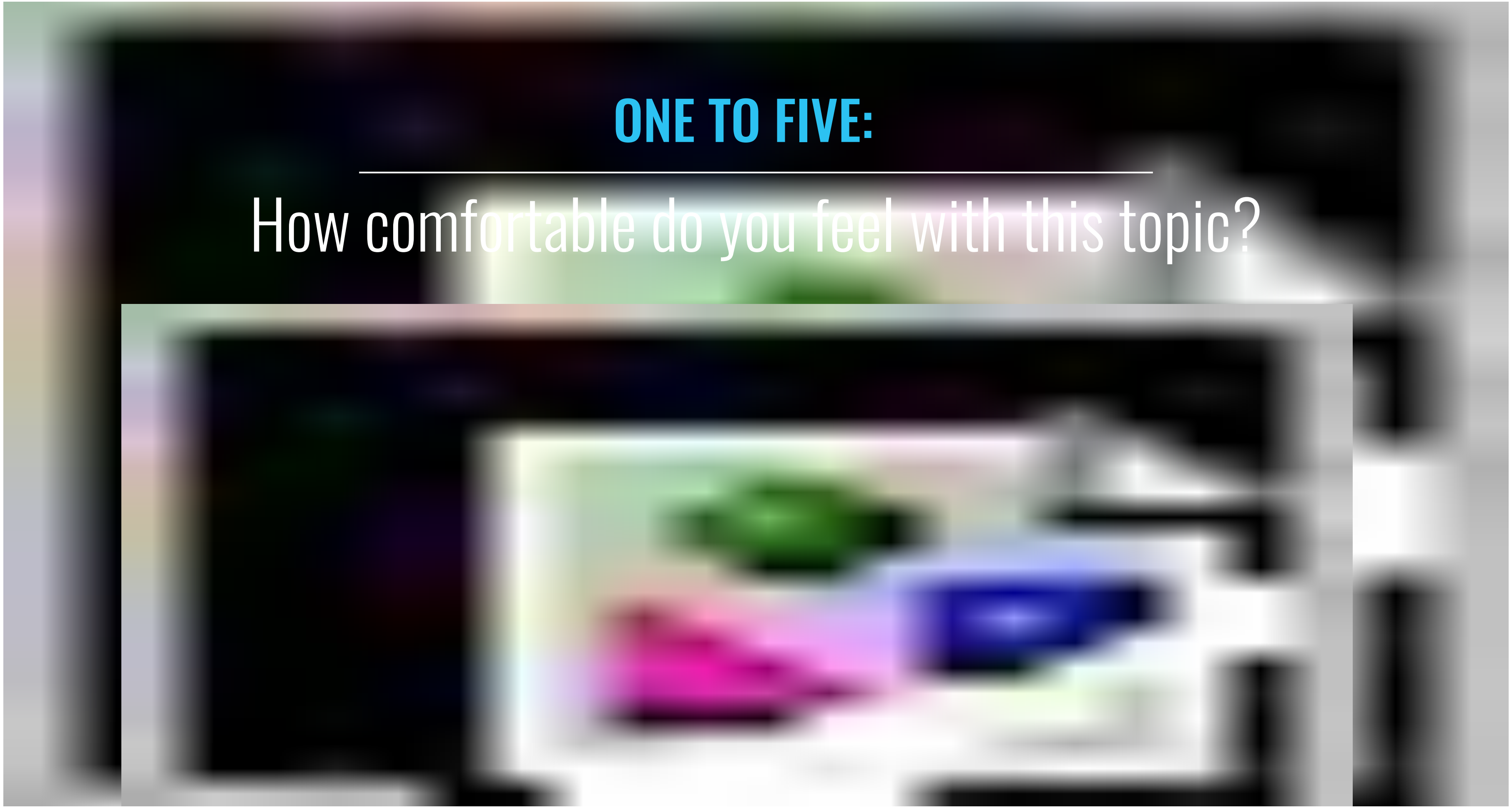
02 Joins

03 Queries

Make sure you’ve downloaded any relevant class files!

ONE TO FIVE:

How comfortable do you feel with this topic?



Time to Code

Installation Check

Suggested Time:

5 minutes

Introduction to SQL

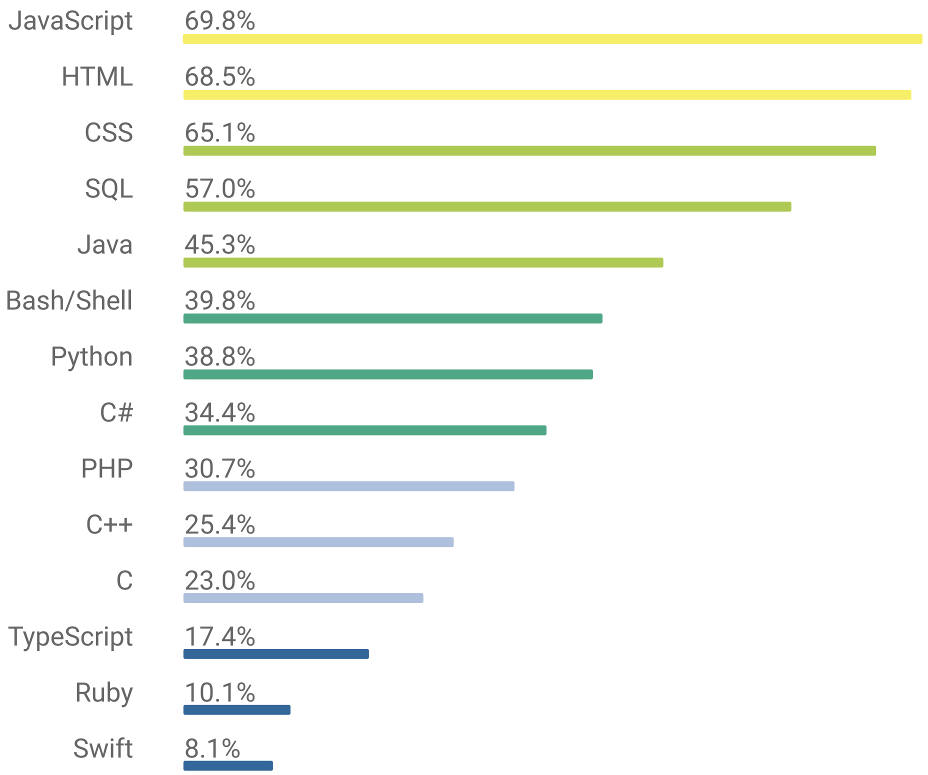
SQL is a powerful programming tool designed to allow programmers to create, populate, manipulate, and access databases, while also providing an easy method for dealing with server-side storage.

Why SQL

Structured **Q**uery **L**anguage (SQL) is one of the main query languages used to access data within relational databases.

SQL is designed to efficiently handle large amounts of data, resulting in high value to organizations.

Experienced **SQL** programmers are in high demand.



Data using SQL is stored in tables on the server, much like spreadsheets you would create in Microsoft Excel. This makes the data easy to visualize and search.

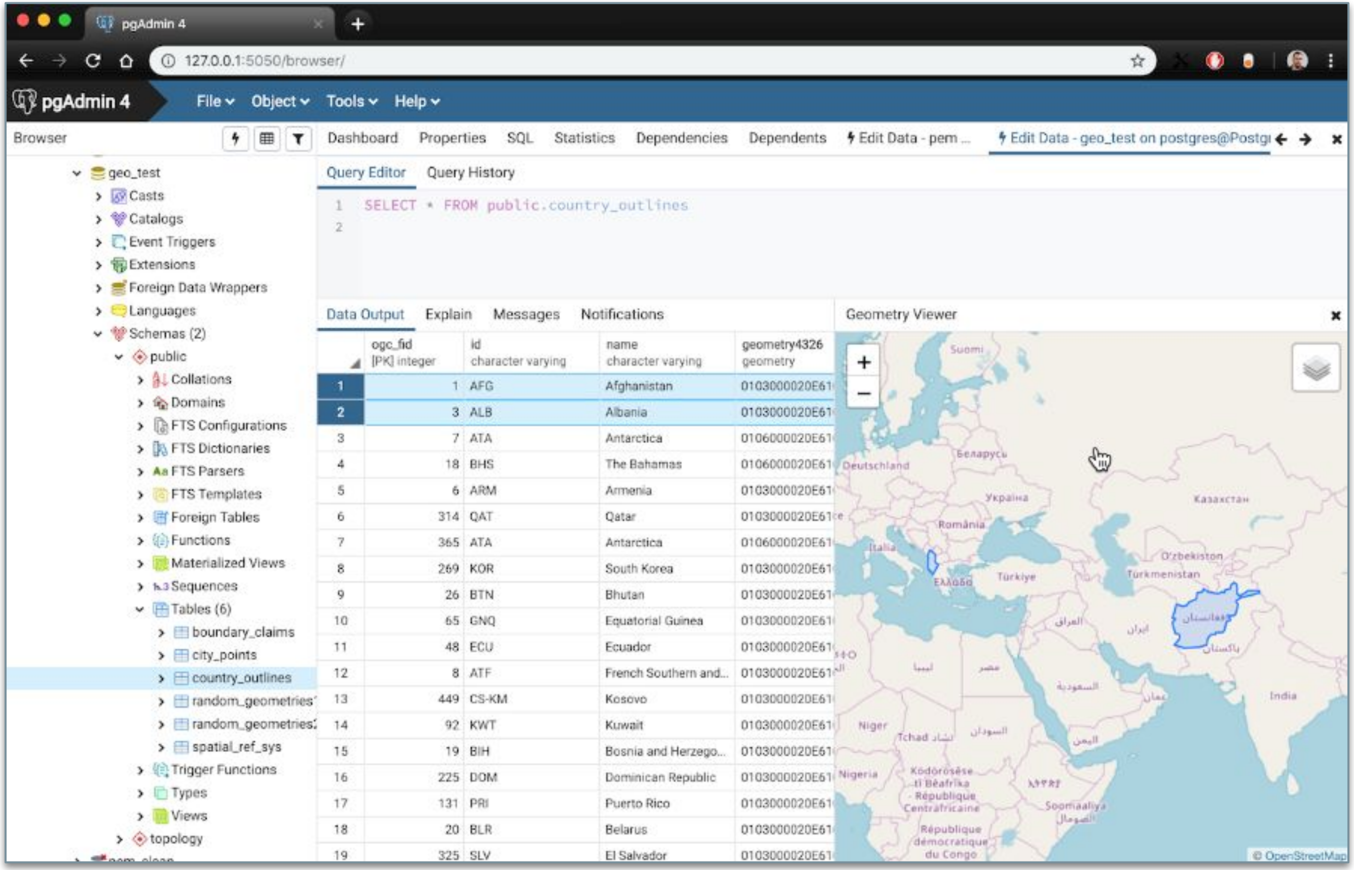
PostgreSQL, usually referred to as "Postgres", is an object-relational database system that uses the SQL language.

PostgreSQL

- Database Engine
- Open Source
- Great Functionality

pgAdmin is the management tool used for working with Postgres.
It simplifies creation, maintenance, and use of database objects

pgAdmin



```
SELECT * FROM public.country_outlines
```

	ogc_fid	id	name	geometry4326
	[PK] integer	character varying	character varying	geometry
1	1	AFG	Afghanistan	0103000020E61
2	3	ALB	Albania	0103000020E61
3	7	ATA	Antarctica	0106000020E61
4	18	BHS	The Bahamas	0106000020E61
5	6	ARM	Armenia	0103000020E61
6	314	QAT	Qatar	0103000020E61
7	365	ATA	Antarctica	0106000020E61
8	269	KOR	South Korea	0103000020E61
9	26	BTN	Bhutan	0103000020E61
10	65	GNQ	Equatorial Guinea	0103000020E61
11	48	ECU	Ecuador	0103000020E61
12	8	ATF	French Southern and...	0103000020E61
13	449	CS-KM	Kosovo	0103000020E61
14	92	KWT	Kuwait	0103000020E61
15	19	BIH	Bosnia and Herzego...	0103000020E61
16	225	DOM	Dominican Republic	0103000020E61
17	131	PRI	Puerto Rico	0103000020E61
18	20	BLR	Belarus	0103000020E61
19	325	SLV	El Salvador	0103000020E61

Creating a Database and Tables

Time to Code

Create a Database

Suggested Time:

5 minutes

Create a Database

Instructions

In the pgAdmin editor, right-click the newly established server to create a new database.

From the menu, select **Create**, and then select **Database** to create a new database

Enter `animals_db` as the database name. Make sure the owner is set as the default postgres, and then click Save.

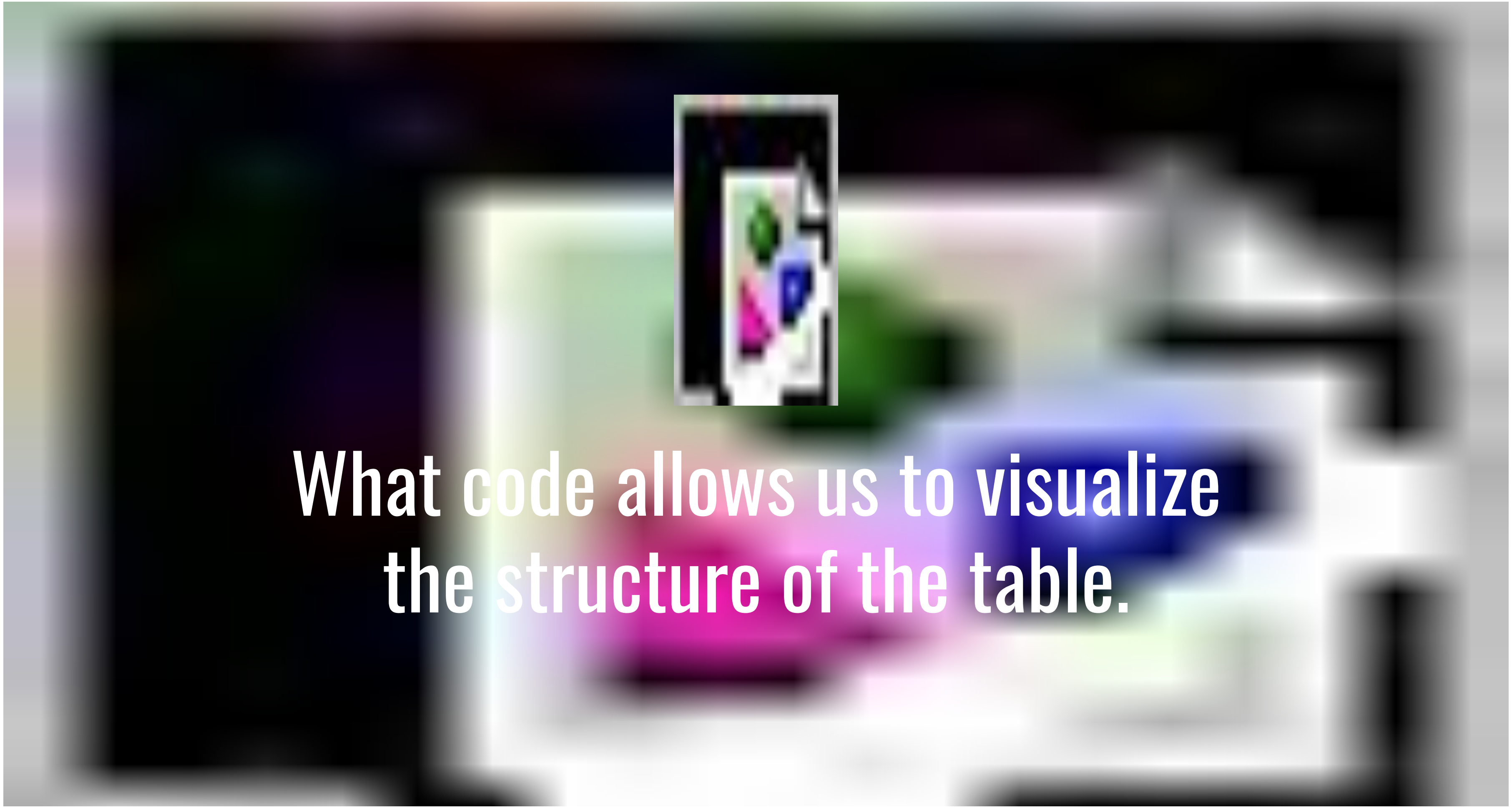
Questions?

Time to Code

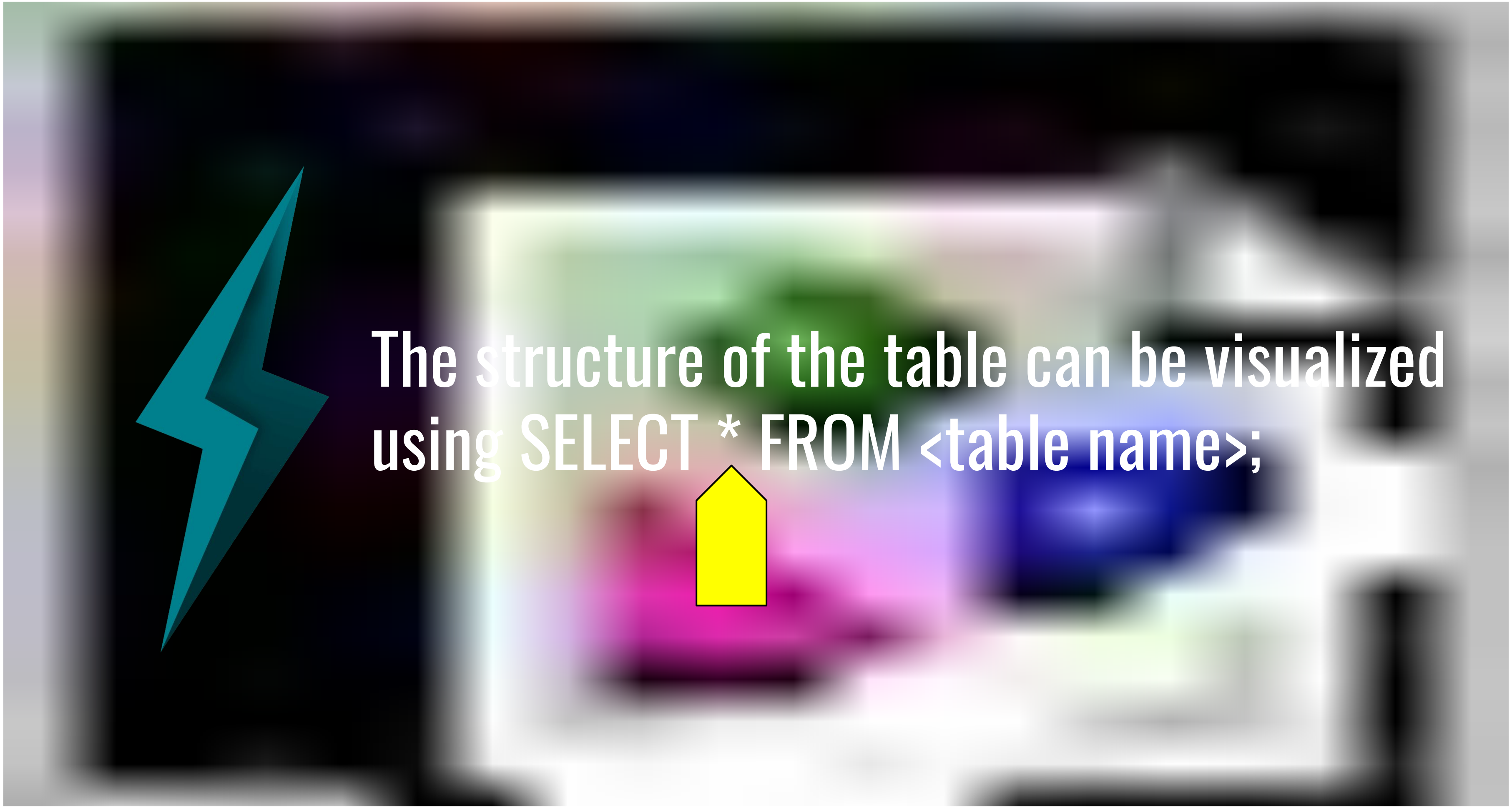
Create Tables

Suggested Time:

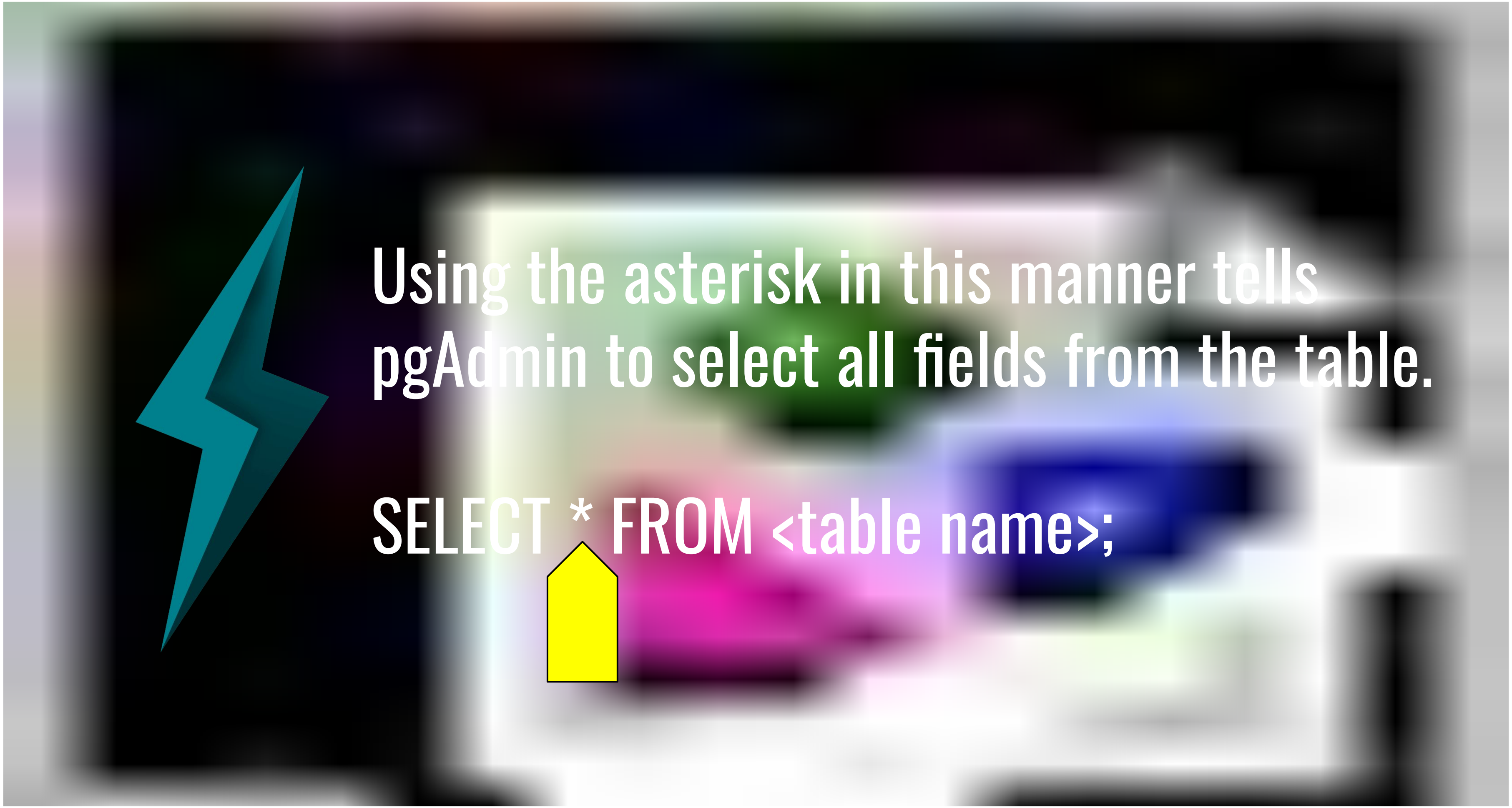
10 minutes



What code allows us to visualize the structure of the table.



The structure of the table can be visualized using SELECT * FROM <table name>;



Using the asterisk in this manner tells pgAdmin to select all fields from the table.

```
SELECT * FROM <table name>;
```

SQL Data

SQL data is persistent

SQL data is persistent; it is not deleted or overwritten when identical commands are run unless specifically commanded.

error message

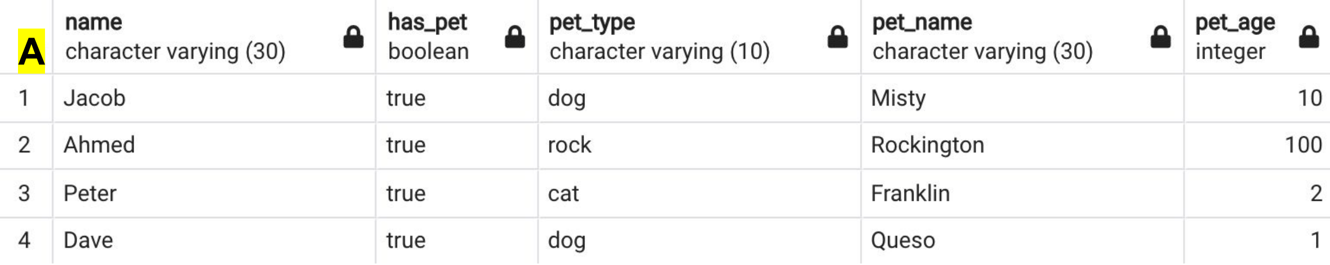
This means that if you try to create a database or table with a name identical to one that already exists, an error will occur telling the user that the database or table already exists.

Instructor Demonstration

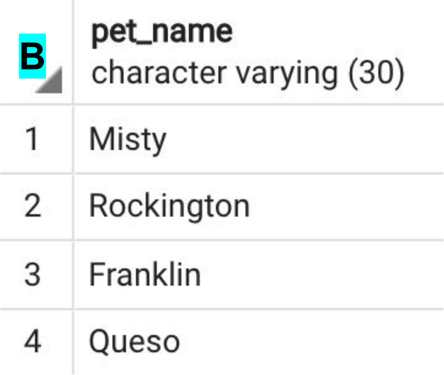
Already Exists Error

Activity: Creating Tables - people

- Create a new database in pgAdmin named `animals_db`.
- Using the query tool, create an empty table named `people`. Be sure to match the data types!
- Insert data into the new table.
- Write a query to view all the data. The result should match table **A**.
- Write a query to view the data from the “pet_name” column. The results should match table **B**.



A	name character varying (30)	has_pet boolean	pet_type character varying (10)	pet_name character varying (30)	pet_age integer
1	Jacob	true	dog	Misty	10
2	Ahmed	true	rock	Rockington	100
3	Peter	true	cat	Franklin	2
4	Dave	true	dog	Queso	1



B	pet_name character varying (30)
1	Misty
2	Rockington
3	Franklin
4	Queso

Querying for Data

01

The SELECT clause can specify more than one column.

02

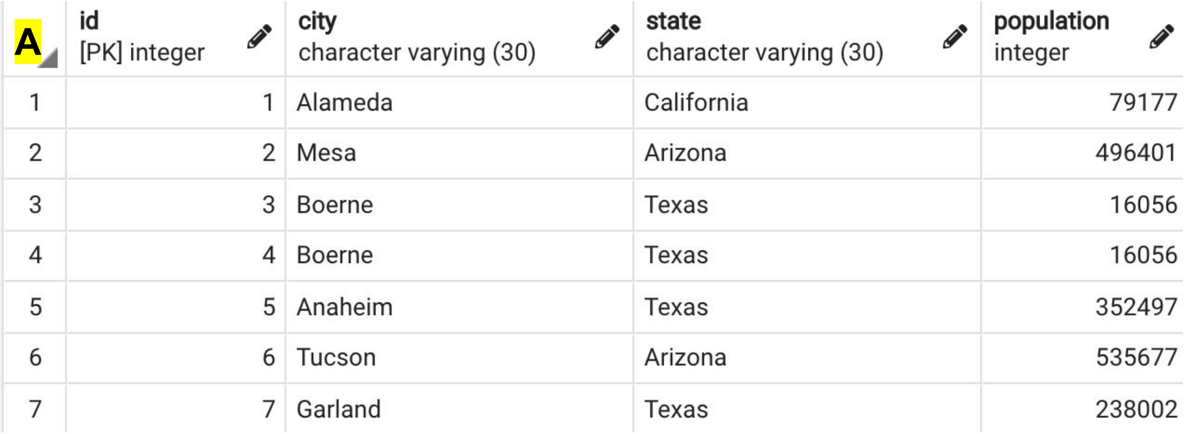
Data is filtered by using additional clauses such as WHERE and AND.

03

The WHERE clause will extract only the data that meets the condition specified. AND adds a second condition to the original clause, further refining the query.

Activity: Creating Tables - cities

- Create a new database in pgAdmin named `city_info`.
- Using the query tool, create an empty table named `cities`. Be sure to match the data types!
- Insert data into the new table.
- Write a query to view all the data. The result should match table **A**.
- Write a query to view the data from the "city" column. The results should match table **B**.



A	id [PK] integer	city character varying (30)	state character varying (30)	population integer
1	1	Alameda	California	79177
2	2	Mesa	Arizona	496401
3	3	Boerne	Texas	16056
4	4	Boerne	Texas	16056
5	5	Anaheim	Texas	352497
6	6	Tucson	Arizona	535677
7	7	Garland	Texas	238002

Activity: Creating Tables

BONUS

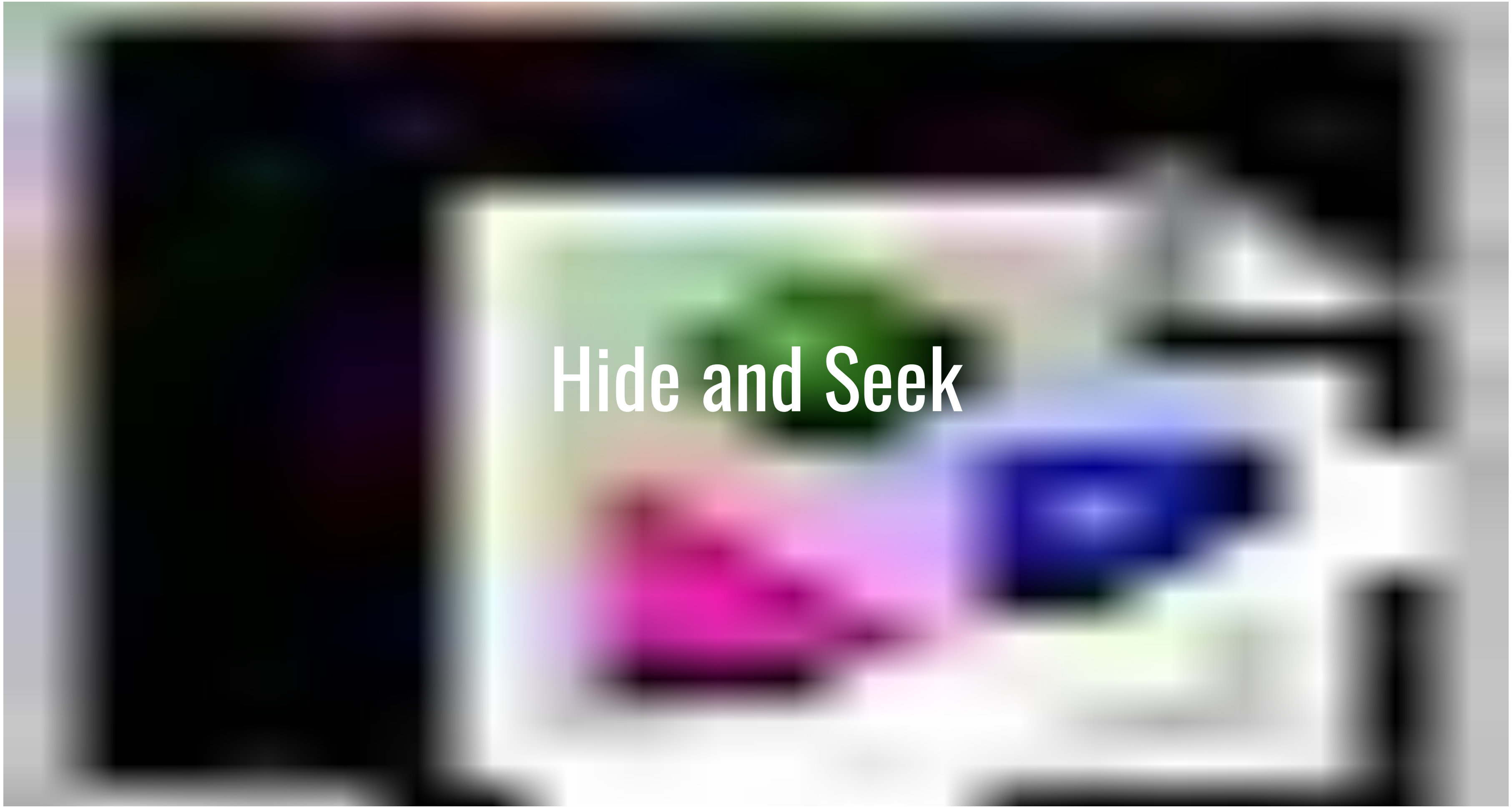
- Filter the table to view only the cities in Texas.
- Filter the table to view only the cities with a population of less than 100,000.
- Filter the table to view California cities with a population of less than 100,000.
- Remove the duplicate entry for Boerne, Texas with the “id” of 4.

HINTS

- For the second bonus question, you will need to use a **WHERE clause** to filter the original query.
- For the third bonus question, an **AND clause** will also be necessary.

Time’s Up! Let’s Review.

Questions?



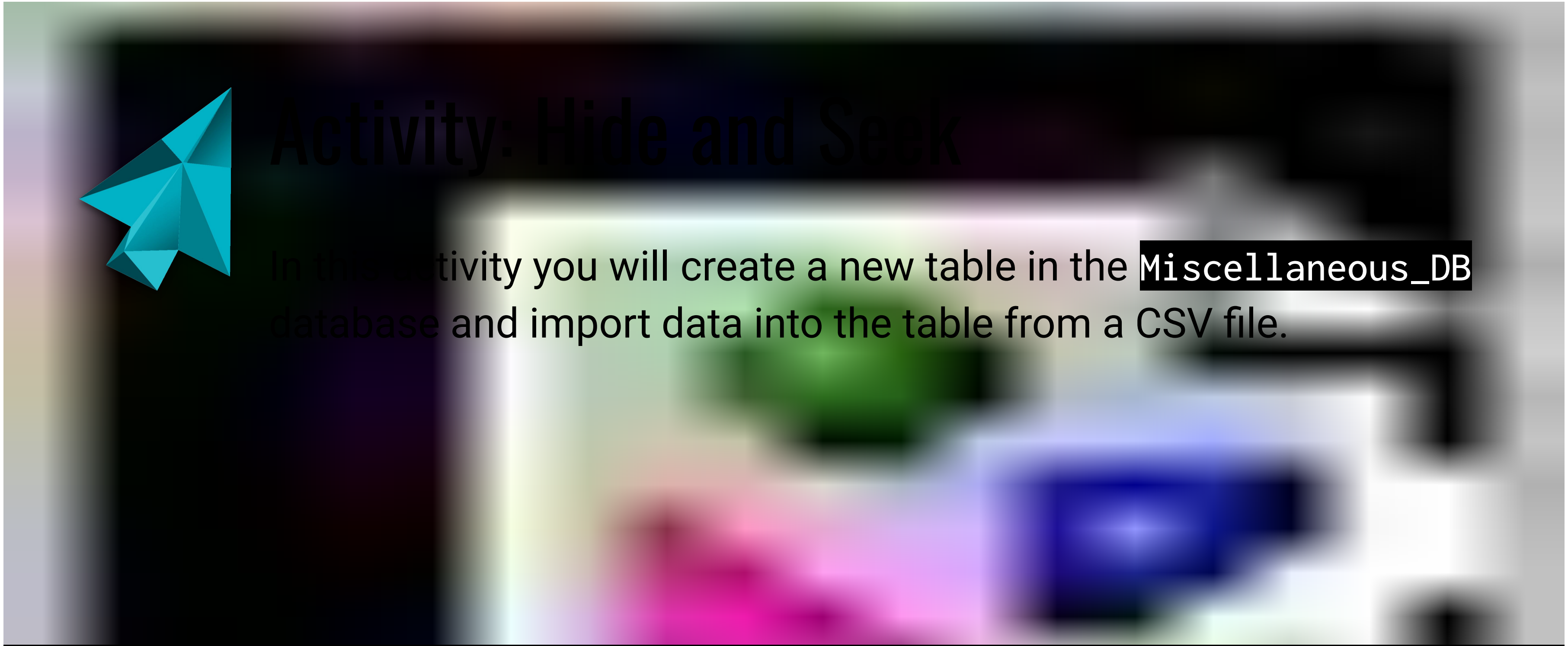
Hide and Seek

Instructor Demonstration

Import Data

Questions?

Activity: Hide and Seek



In this activity you will create a new table in the `Miscellaneous_DB` database and import data into the table from a CSV file.

Suggested Time:

15 minutes

Activity: Hide and Seek

Instructions:

- Create a new table in the `Miscellaneous_DB` database called `wordassociation`.
- Import the data from the `wordassociation_AC.csv` file in the Resources folder.
- Create a query in which the data in the `word1` column is `stone`.
- Create a query that collects all rows in which the author is within the range 0–10.
- Create a query that searches for any rows that have `pie` in their `word1` or `word2` columns.

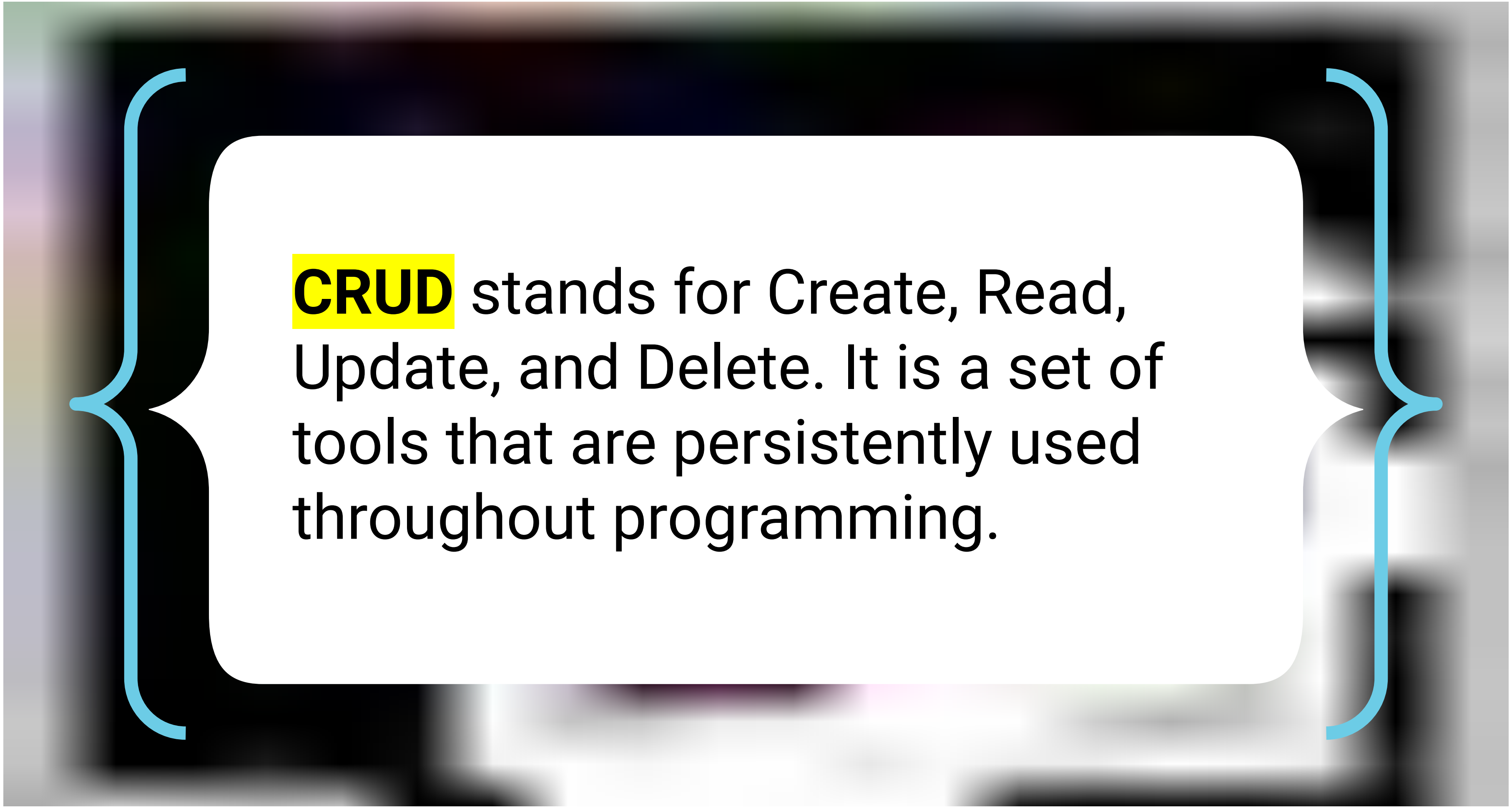
Bonus

- Import to the `wordassociation_AC.csv` to the `wordassociation` table explore filtering on the `source` column.
- Create a query that will collect all rows with a `source` of BC.
- Create a query that will collect all rows with a `source` of BC and an author range between 333 and 335.

Note: Data provided by Kaggle.

Time’s Up! Let’s Review.

Using CRUD

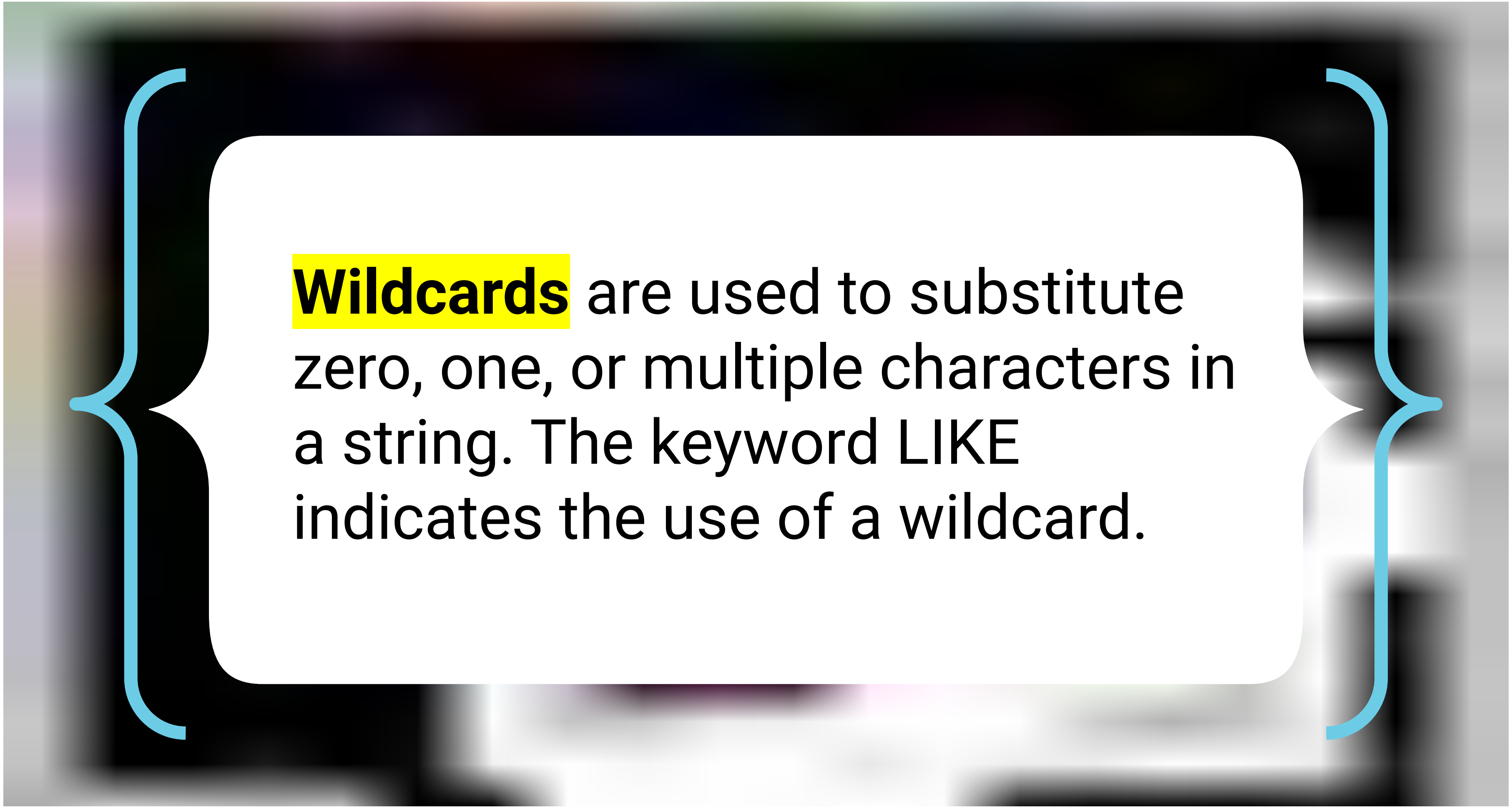


CRUD stands for Create, Read, Update, and Delete. It is a set of tools that are persistently used throughout programming.

CRUD

Create	`INSERT table info (column1, column2, column3)`
Read	`SELECT * FROM table`
Update	`UPDATE table SET column1 = VALUE WHERE id = 1`
Delete	`DELETE FROM table WHERE id = 4`

These tools are fundamental to all programming languages—not just SQL.



Wildcards are used to substitute zero, one, or multiple characters in a string. The keyword LIKE indicates the use of a wildcard.

Wildcard: % and _

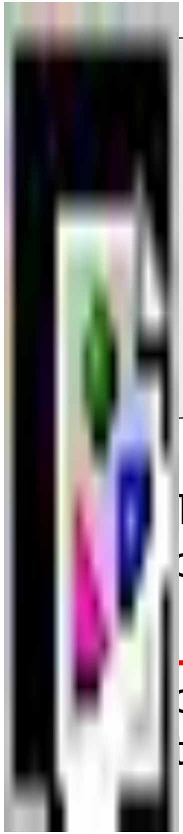
```
SELECT *

FROM actor

WHERE last_name LIKE ‘Will%’;
```

The **%** will substitute **zero**, **one**, or **multiple** characters in a query.

For example, all of the following will match: **Will**, **Willa**, and **Willows**.



```
SELECT *

FROM actor

WHERE first_name LIKE ‘_AN’;
```

The **_** will substitute one, and only one, character in a query.

_AN returns all actors whose first name contains three letters, the second and third of which are **AN**.

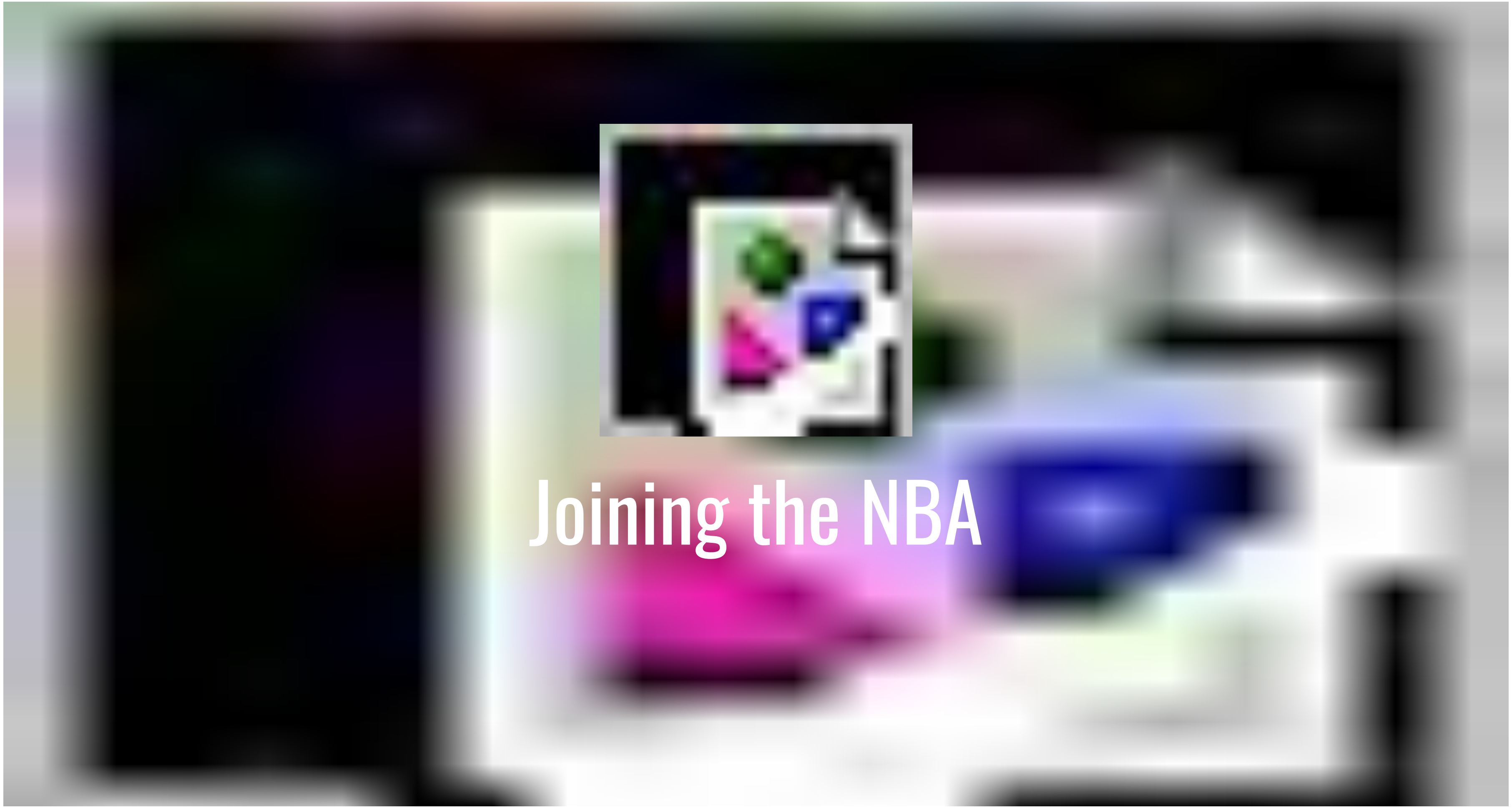
Questions?

Time to Code

Using CRUD

Suggested Time:

20 minutes



Joining the NBA

Instructor Demonstration

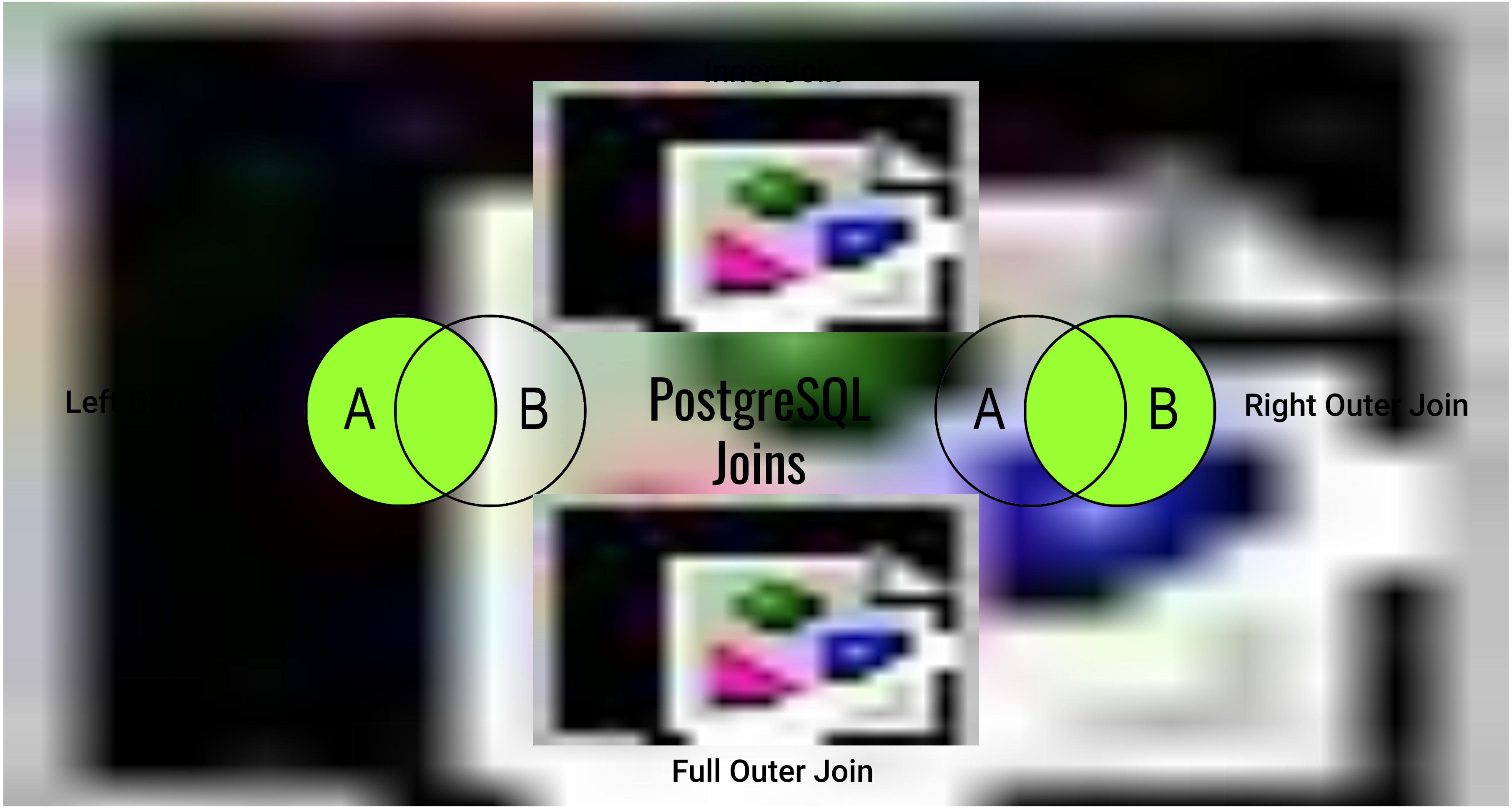
Joins

Joins

Five Primary Types of Joins used with PostgreSQL:

INNER JOIN	returns records that have matching values in both tables.
LEFT JOIN	returns all records from the left table and the matched records from the right table.
RIGHT JOIN	returns all records from the right table and the matched records from the left table.
CROSS JOIN	returns records that match every row of the left table with every row of the right table. This type of join has the potential to make very large tables.
FULL OUTER JOIN	places null values within the columns that do not match between the two tables, after an inner join is performed.

PostgreSQL Joins



Inner Join

Left Outer Join

Right Outer Join

Full Outer Join

Joins

In our given scenario `player_id` column of the players table and the `loser_id/winner_id` columns of the `matches` table have matching values.

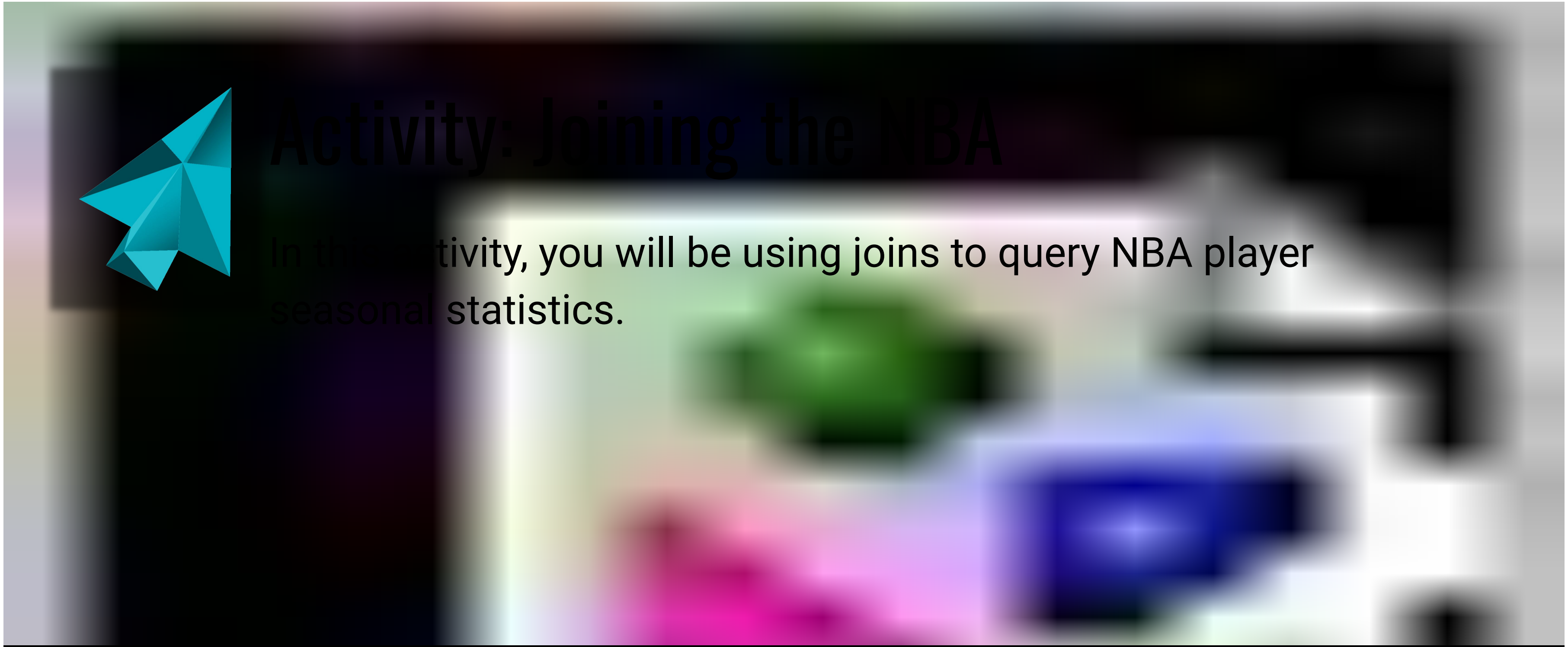
In that case we can join these tables together utilizing the **INNER JOIN**:

```
SELECT players.first_name, players.last_name, players.hand,
matches.loser_rank
FROM matches
INNER JOIN players ON
players.player_id=matches.loser_id;
```

Joins

A more advanced **INNER JOIN** solution.

```sql
-- Advanced INNER JOIN solution
SELECT p.first_name, p.last_name, p.hand, m.loser_rank
FROM matches AS m
INNER JOIN players AS p ON
p.player_id=m.loser_id;
```



Activity: Joining the NBA

In this activity, you will be using joins to query NBA player seasonal statistics.

Suggested Time:

20 minutes

Activity: Joining the NBA

Instructions:

Create a new database named `NBA_DB` and create two new tables with pgAdmin named `players` and `seasons_stats`.

Copy the code from `schema.sql` to create the tables, and then import the corresponding data from `Players.csv` and `Seasons_Stats.csv`.

Remember to refresh the database; newly created tables will not immediately appear.

Activity: Joining the NBA

Perform joins that will generate the following outputs.

Basic Information Table:

Percents Stats:

Time’s Up! Let’s Review.

Questions?