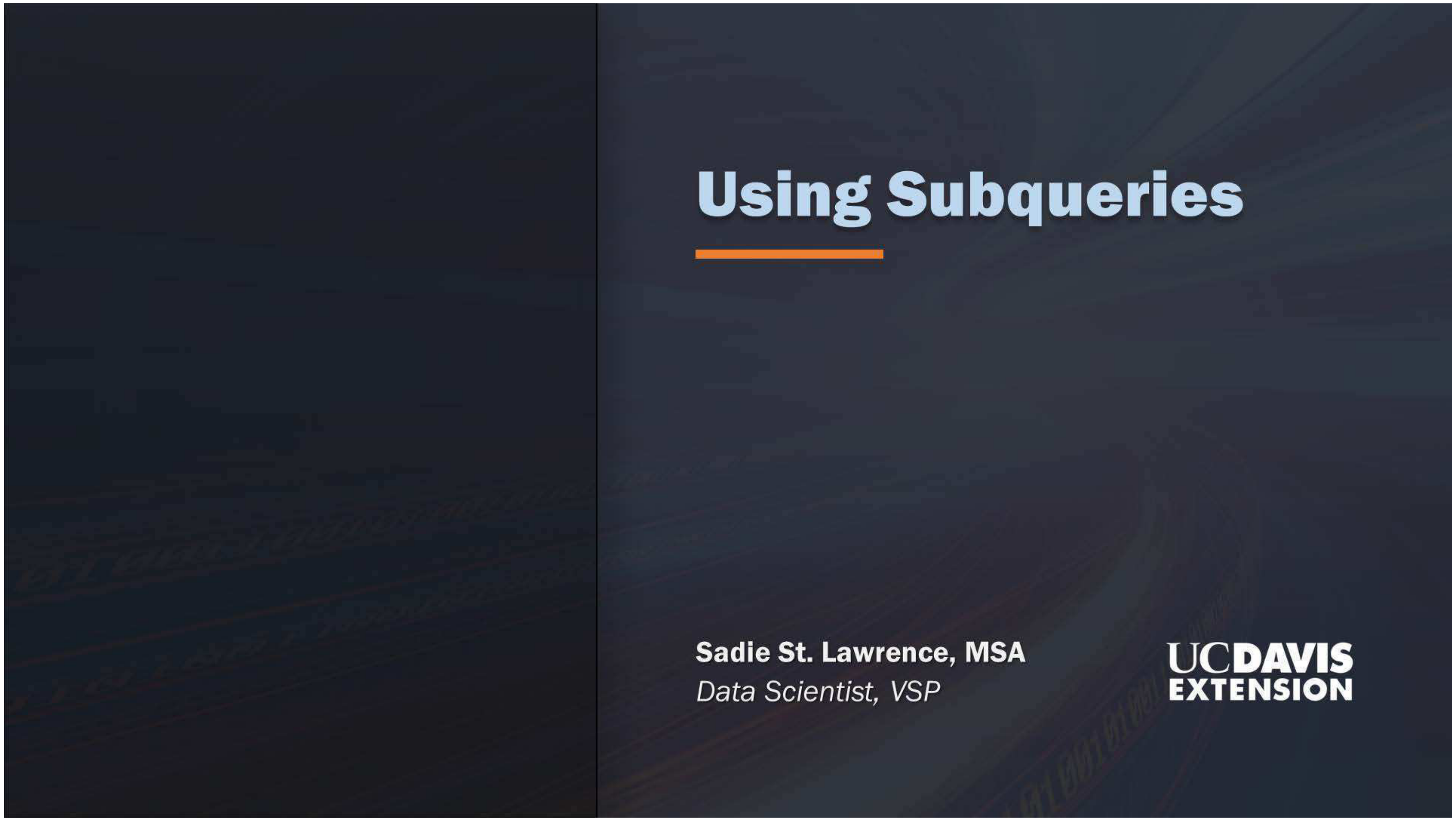
Slide 1: Using Subqueries

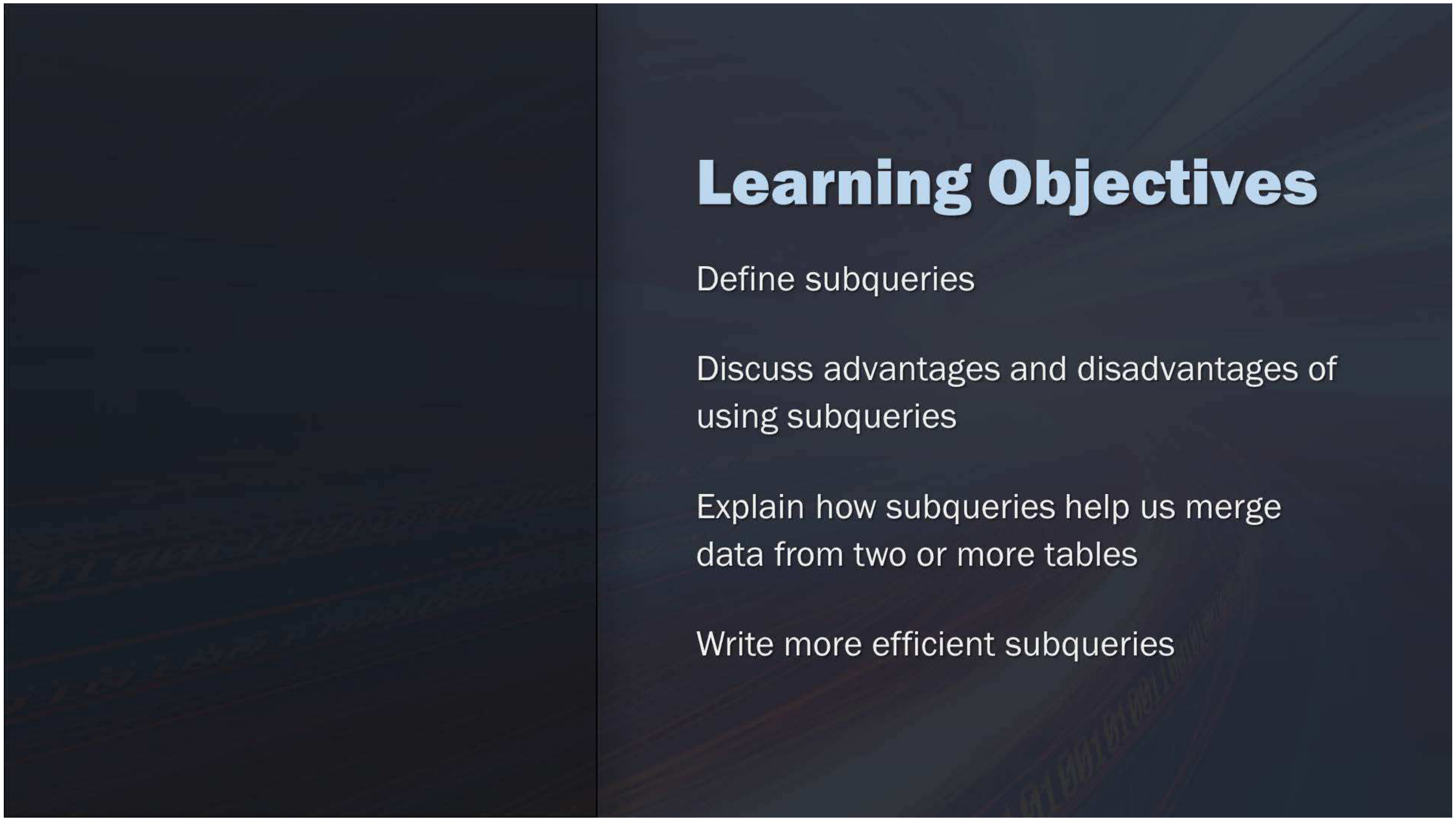


Using Subqueries

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Learning Objectives



Learning Objectives

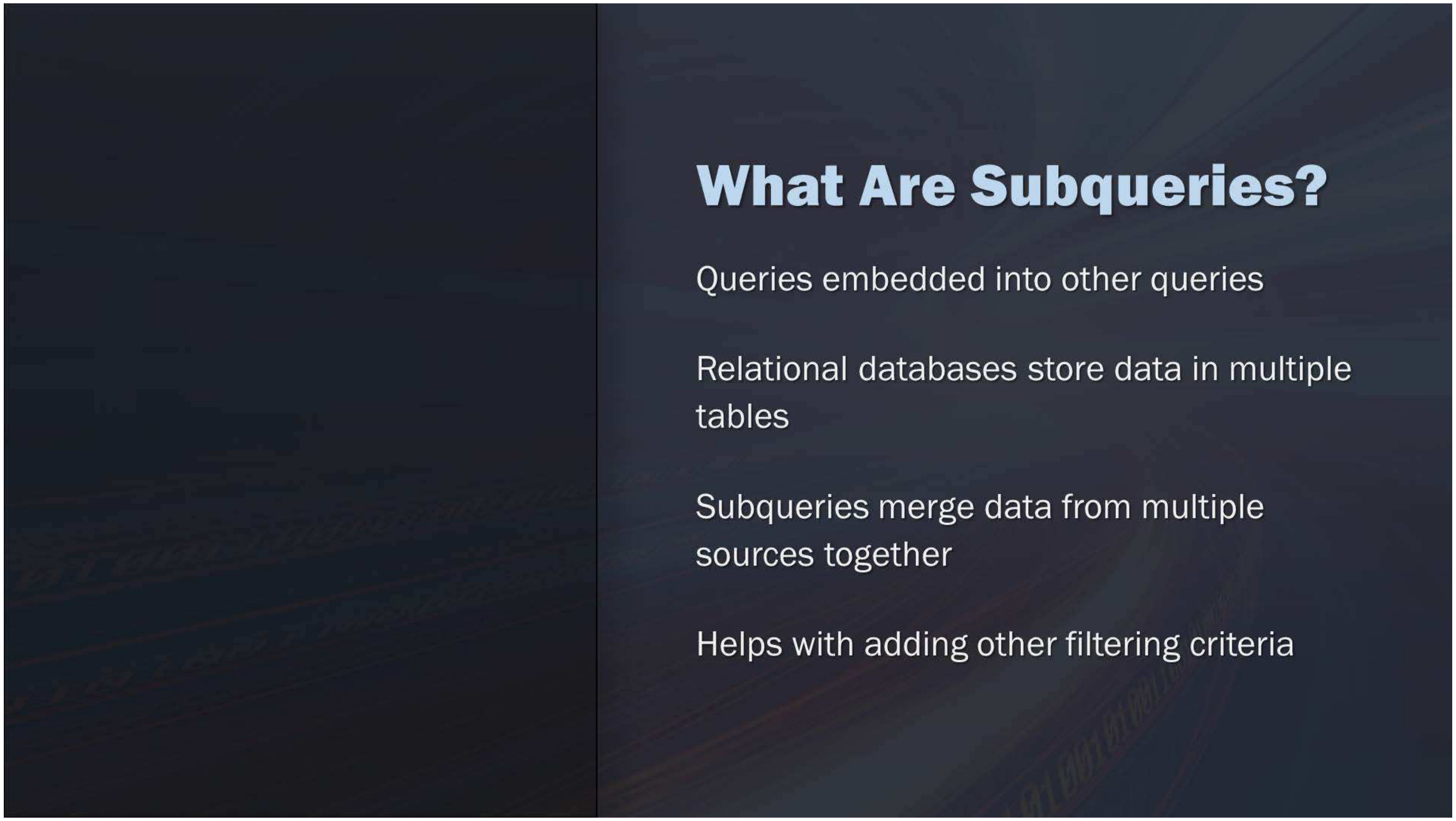
Define subqueries

Discuss advantages and disadvantages of using subqueries

Explain how subqueries help us merge data from two or more tables

Write more efficient subqueries

Slide 3: What Are Subqueries



What Are Subqueries?

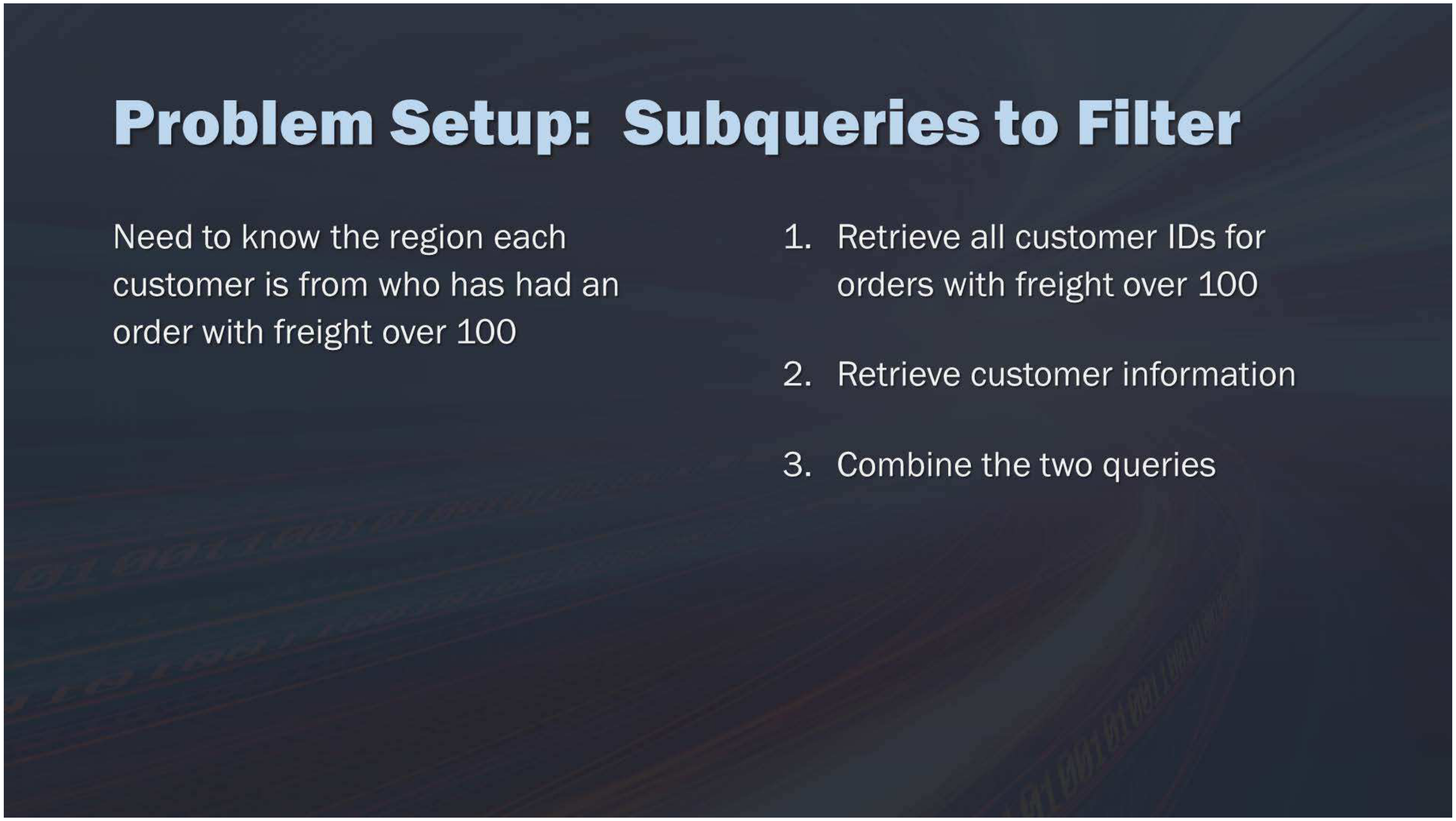
Queries embedded into other queries

Relational databases store data in multiple tables

Subqueries merge data from multiple sources together

Helps with adding other filtering criteria

Slide 4: Problem Setup: Subqueries to Filter

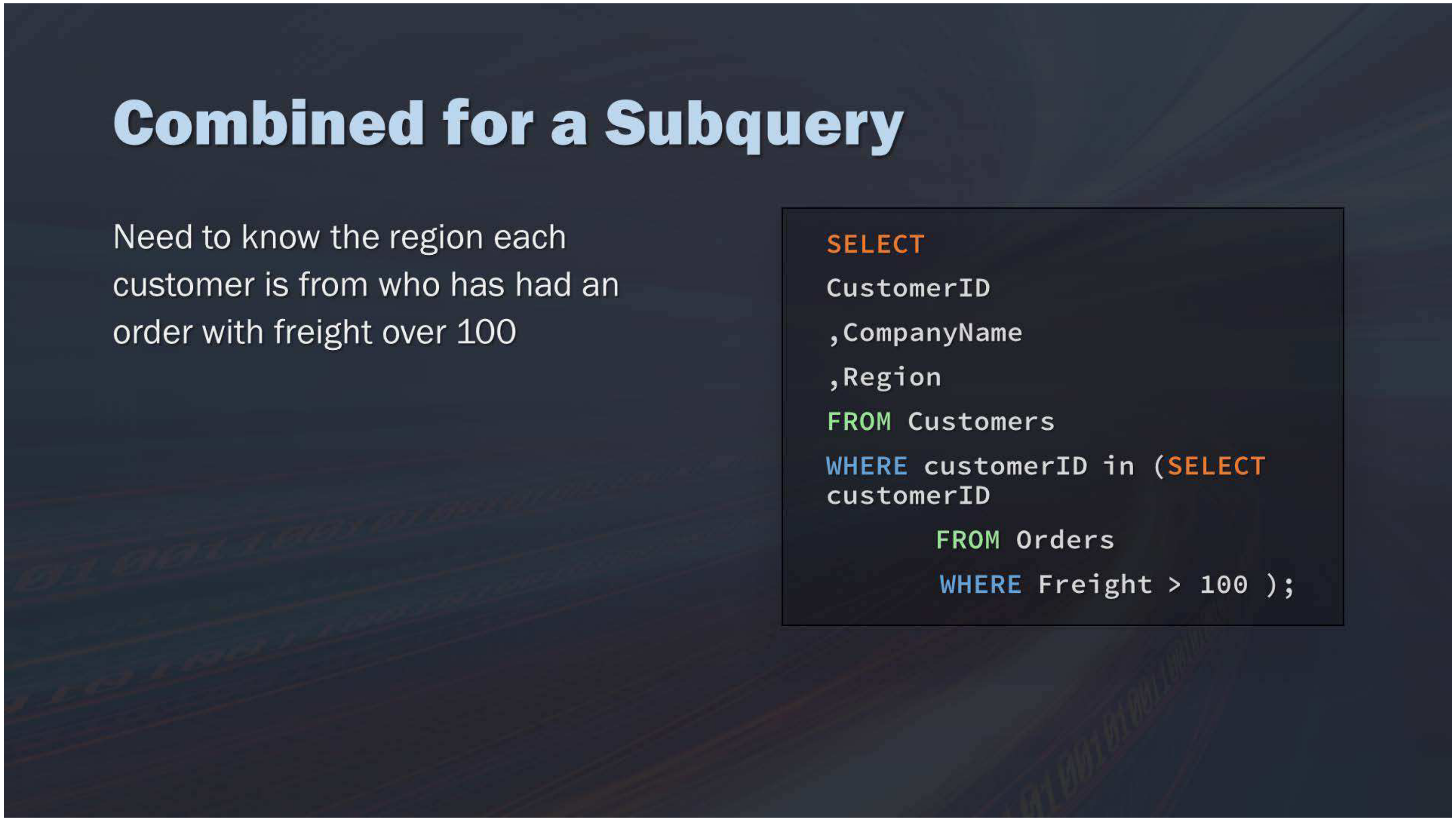


Problem Setup: Subqueries to Filter

Need to know the region each customer is from who has had an order with freight over 100

1. Retrieve all customer IDs for orders with freight over 100
2. Retrieve customer information
3. Combine the two queries

Slide 5: Example

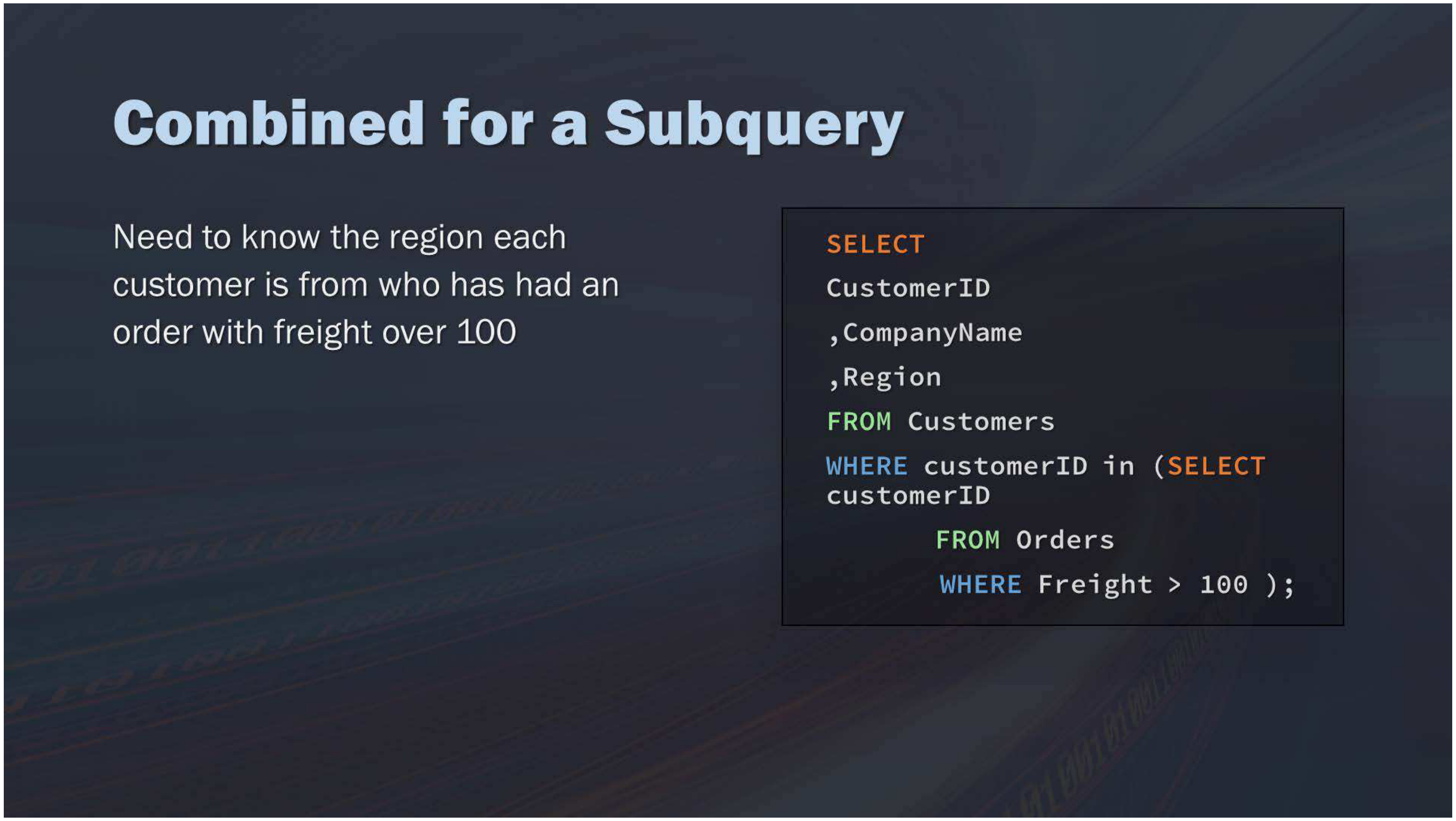


Combined for a Subquery

Need to know the region each customer is from who has had an order with freight over 100

```
SELECT
CustomerID
,CompanyName
,Region
FROM Customers
WHERE customerID in (SELECT
customerID
       FROM Orders
       WHERE Freight > 100 );
```

Slide 6: Combined for a Subquery

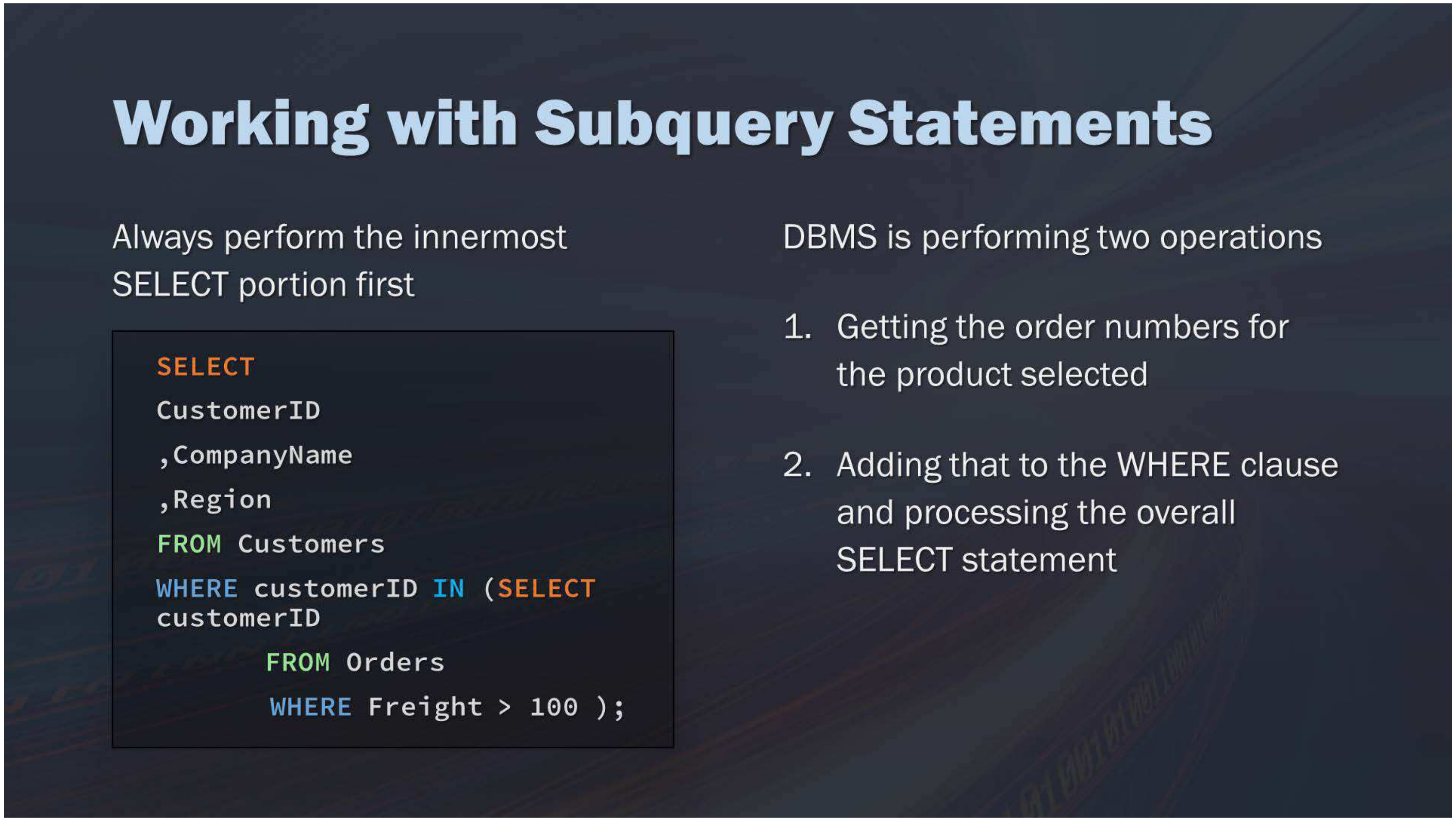


Combined for a Subquery

Need to know the region each customer is from who has had an order with freight over 100

```
SELECT
CustomerID
,CompanyName
,Region
FROM Customers
WHERE customerID in (SELECT
customerID
       FROM Orders
       WHERE Freight > 100 );
```

Slide 7: Working with Subquery Statements



Working with Subquery Statements

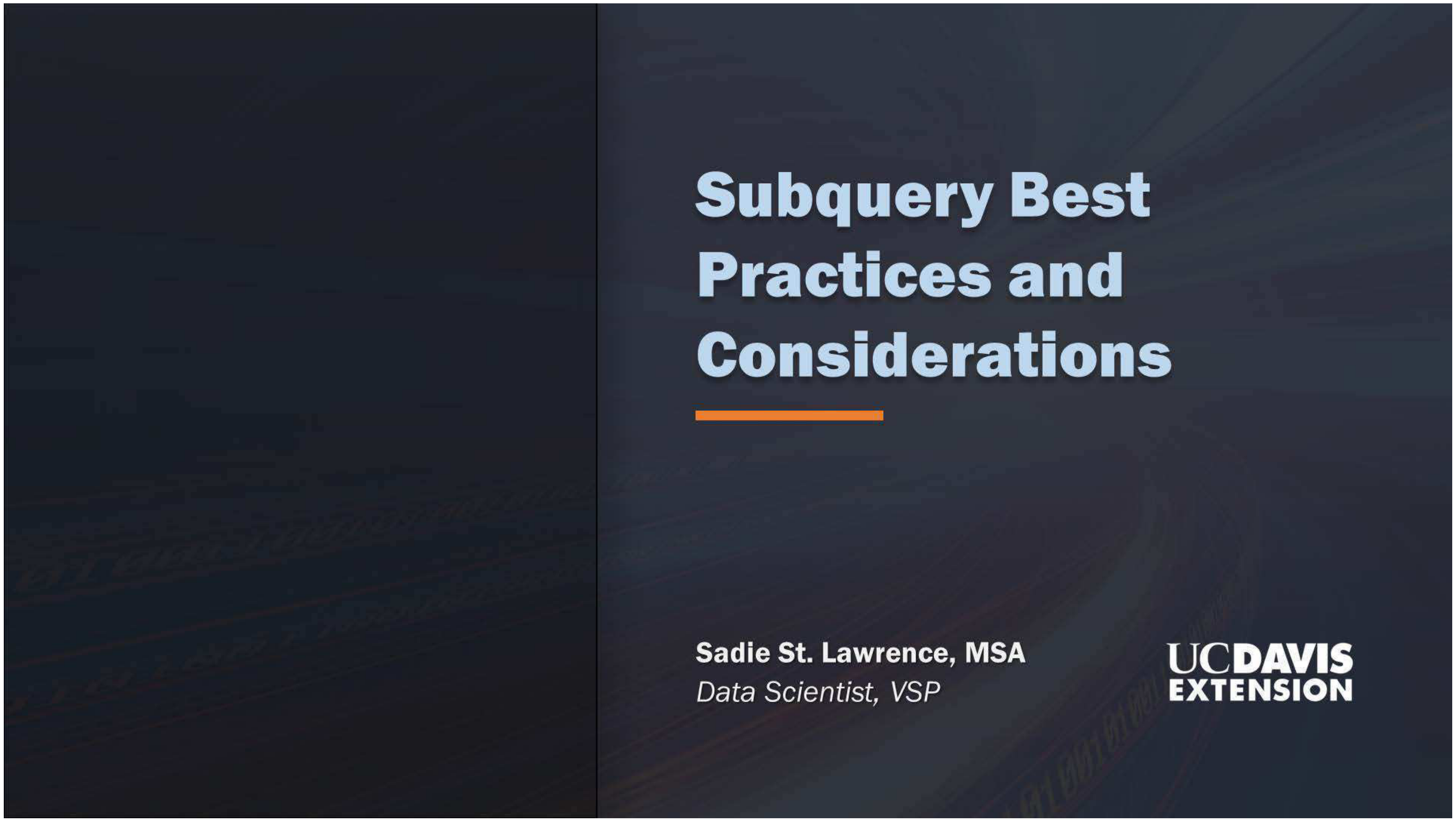
Always perform the innermost SELECT portion first

```
SELECT
CustomerID
,CompanyName
,Region
FROM Customers
WHERE customerID IN (SELECT
customerID
      FROM Orders
      WHERE Freight > 100 );
```

DBMS is performing two operations

1. Getting the order numbers for the product selected
2. Adding that to the WHERE clause and processing the overall SELECT statement

Slide 1: Subquery Best Practices and Considerations

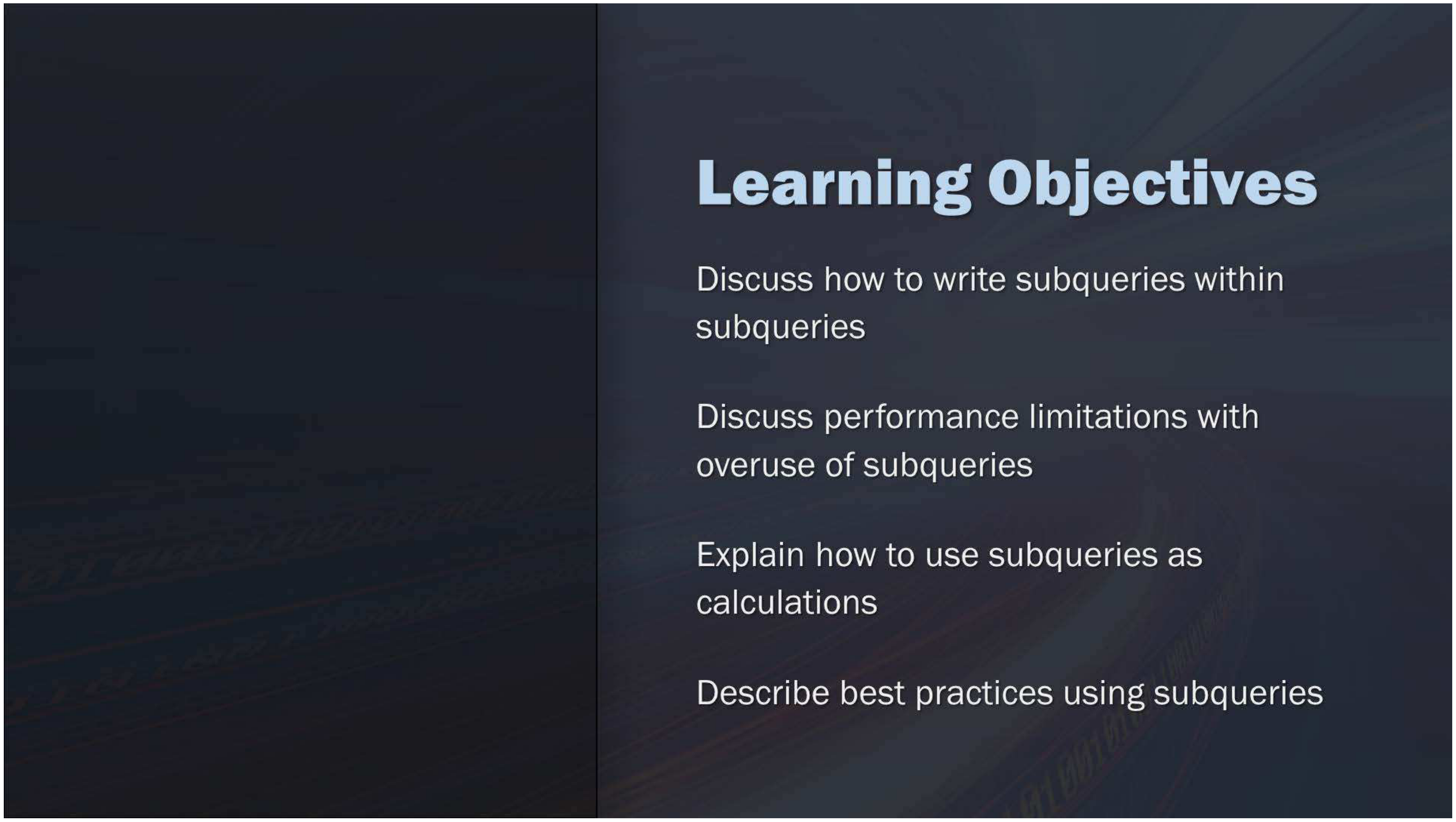


Subquery Best Practices and Considerations

Sadie St. Lawrence, MSA
Data Scientist, VSP

UC DAVIS EXTENSION

Slide 2: Learning Objectives



Learning Objectives

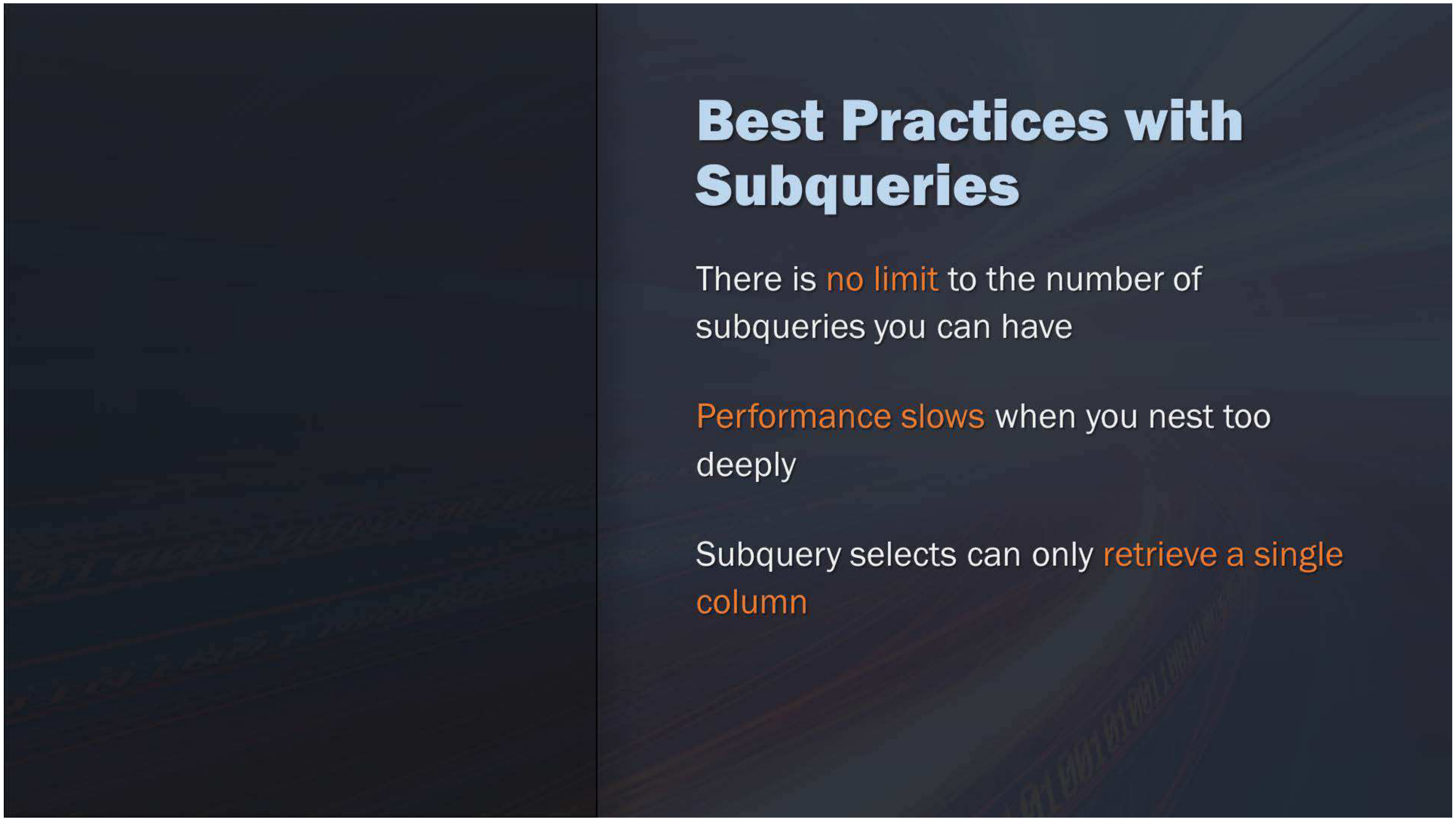
Discuss how to write subqueries within subqueries

Discuss performance limitations with overuse of subqueries

Explain how to use subqueries as calculations

Describe best practices using subqueries

Slide 3: Best Practices With Subqueries



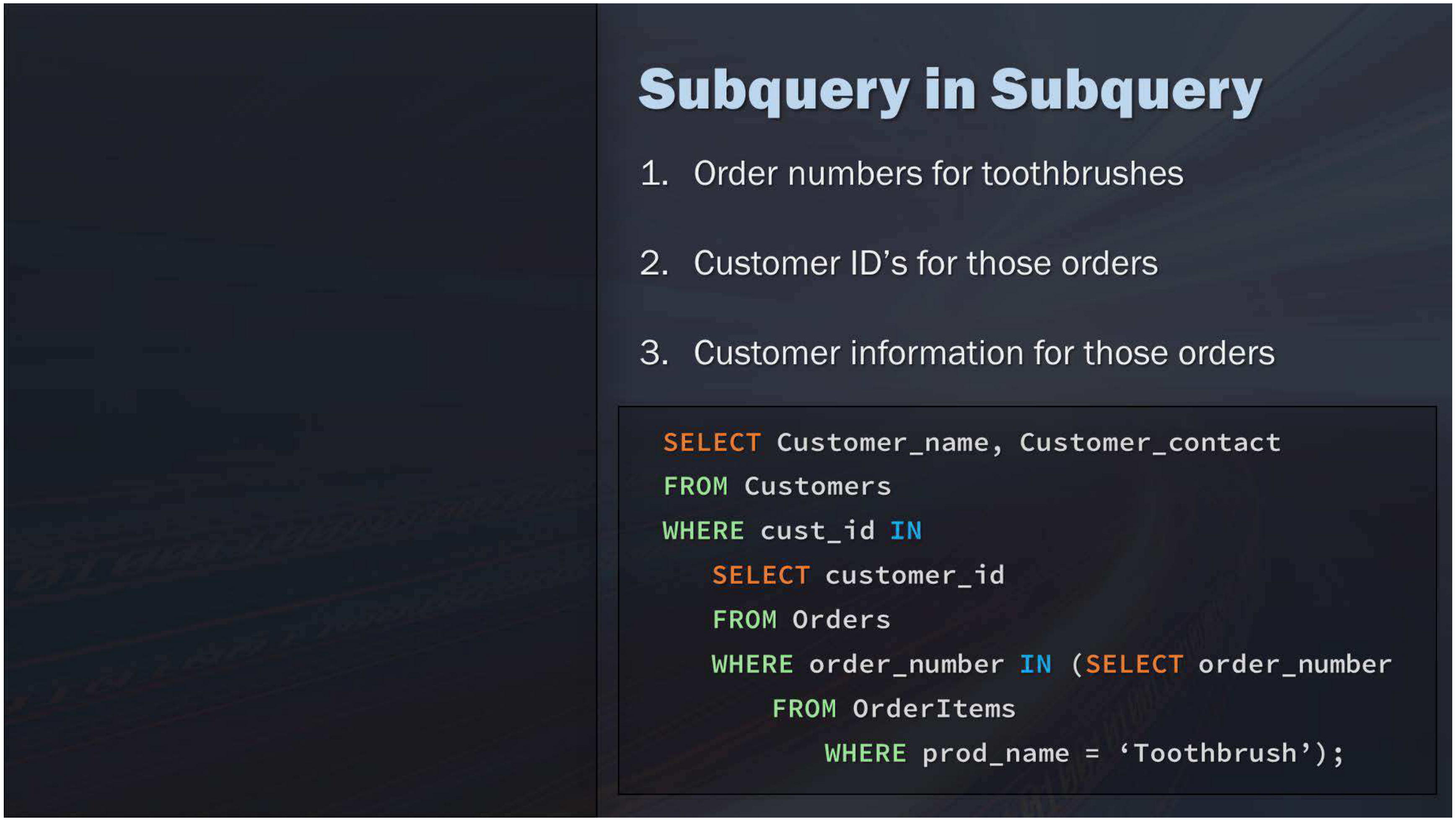
Best Practices with Subqueries

There is no limit to the number of subqueries you can have

Performance slows when you nest too deeply

Subquery selects can only retrieve a single column

Slide 4: Subquery in Subquery

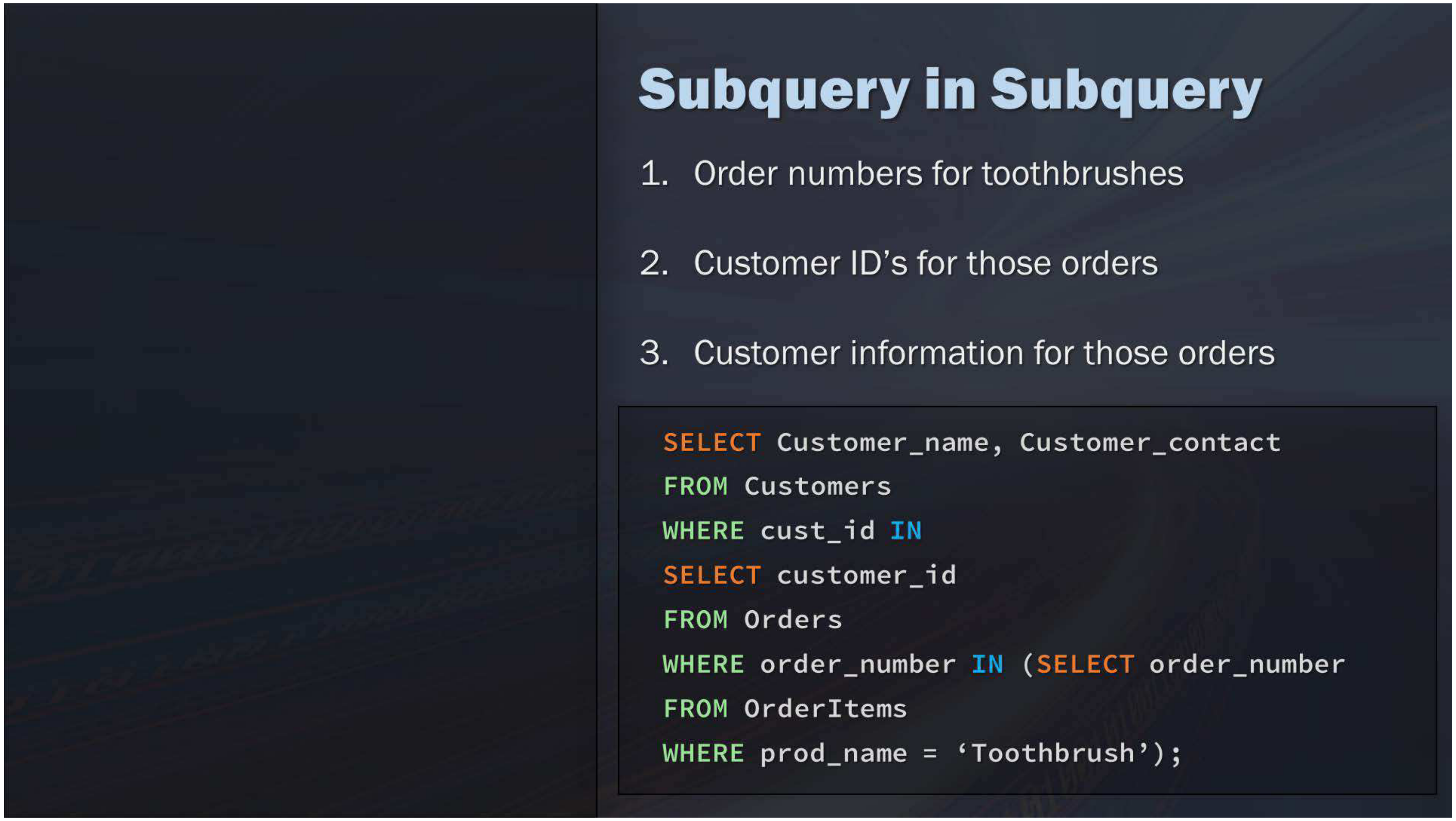


Subquery in Subquery

1. Order numbers for toothbrushes
2. Customer ID's for those orders
3. Customer information for those orders

```sql
SELECT Customer_name, Customer_contact
FROM Customers
WHERE cust_id IN
   SELECT customer_id
   FROM Orders
   WHERE order_number IN (SELECT order_number
       FROM OrderItems
          WHERE prod_name = 'Toothbrush');
```

Slide 5: Subquery in Subquery

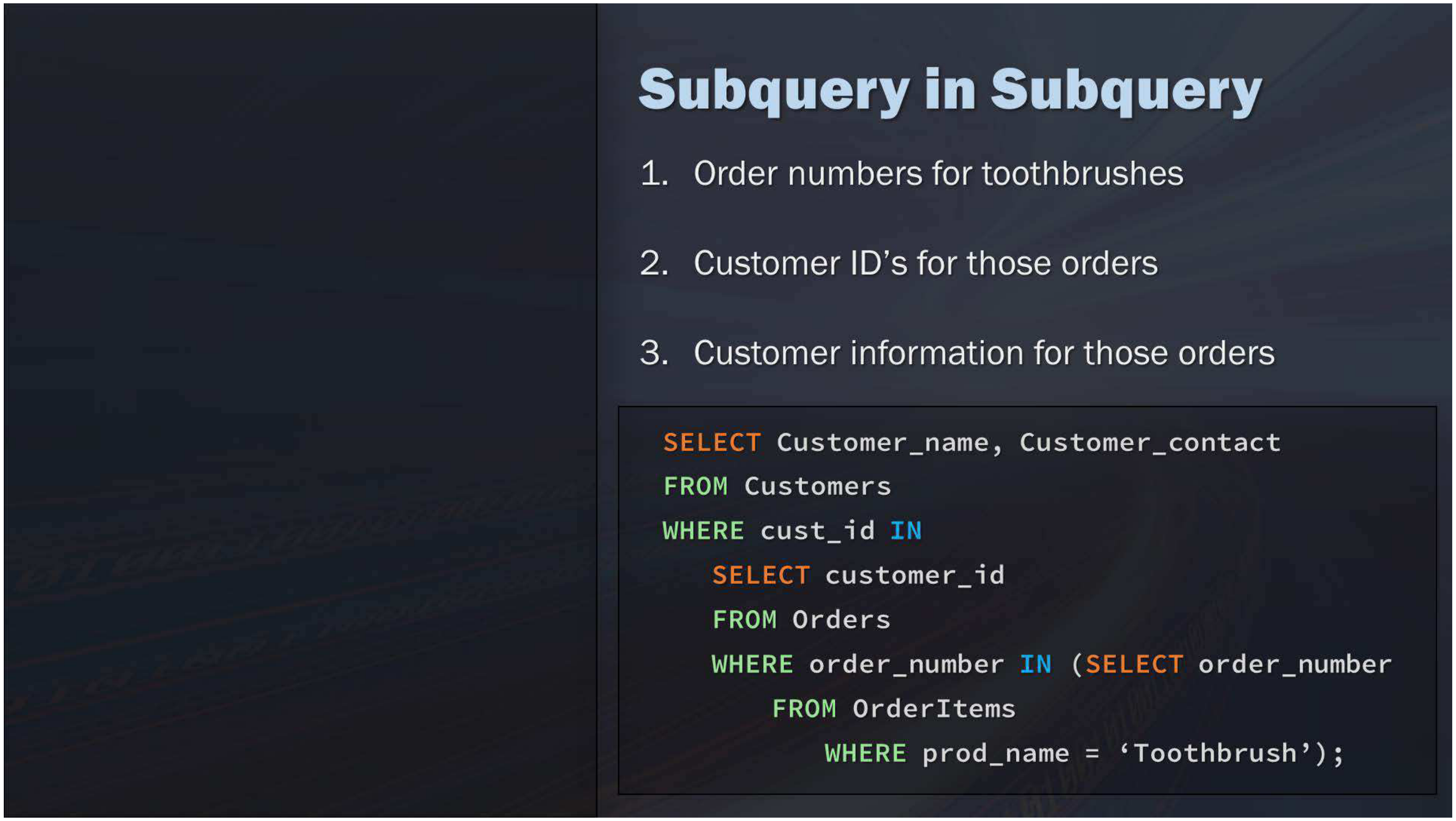


Subquery in Subquery

1. Order numbers for toothbrushes
2. Customer ID's for those orders
3. Customer information for those orders

```
SELECT Customer_name, Customer_contact
FROM Customers
WHERE cust_id IN
SELECT customer_id
FROM Orders
WHERE order_number IN (SELECT order_number
FROM OrderItems
WHERE prod_name = 'Toothbrush');
```

Slide 6: Subquery in Subquery

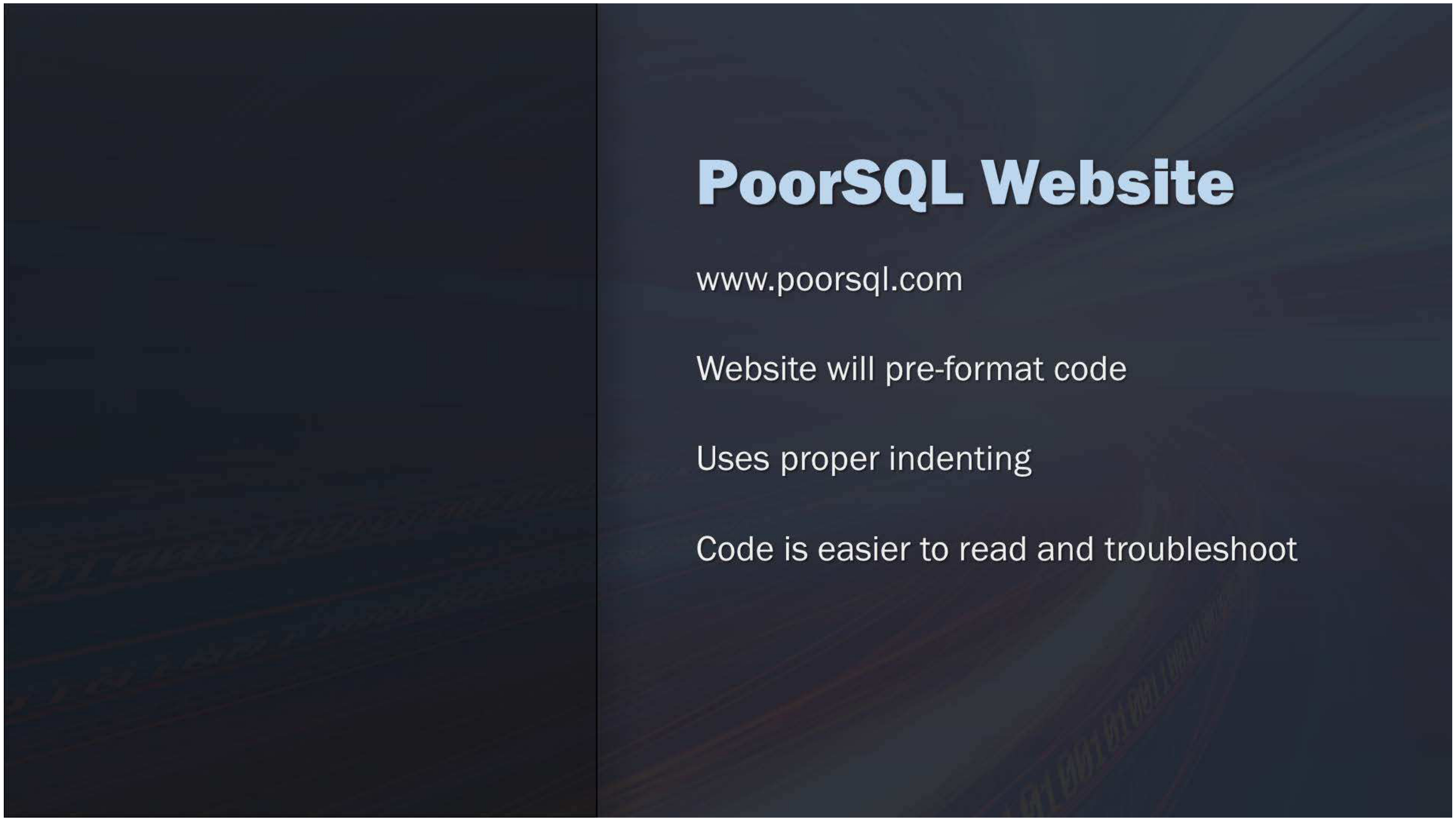


Subquery in Subquery

1. Order numbers for toothbrushes
2. Customer ID's for those orders
3. Customer information for those orders

```
SELECT Customer_name, Customer_contact
FROM Customers
WHERE cust_id IN
   SELECT customer_id
   FROM Orders
   WHERE order_number IN (SELECT order_number
       FROM OrderItems
          WHERE prod_name = 'Toothbrush');
```

Slide 7: PoorSQL Website



PoorSQL Website

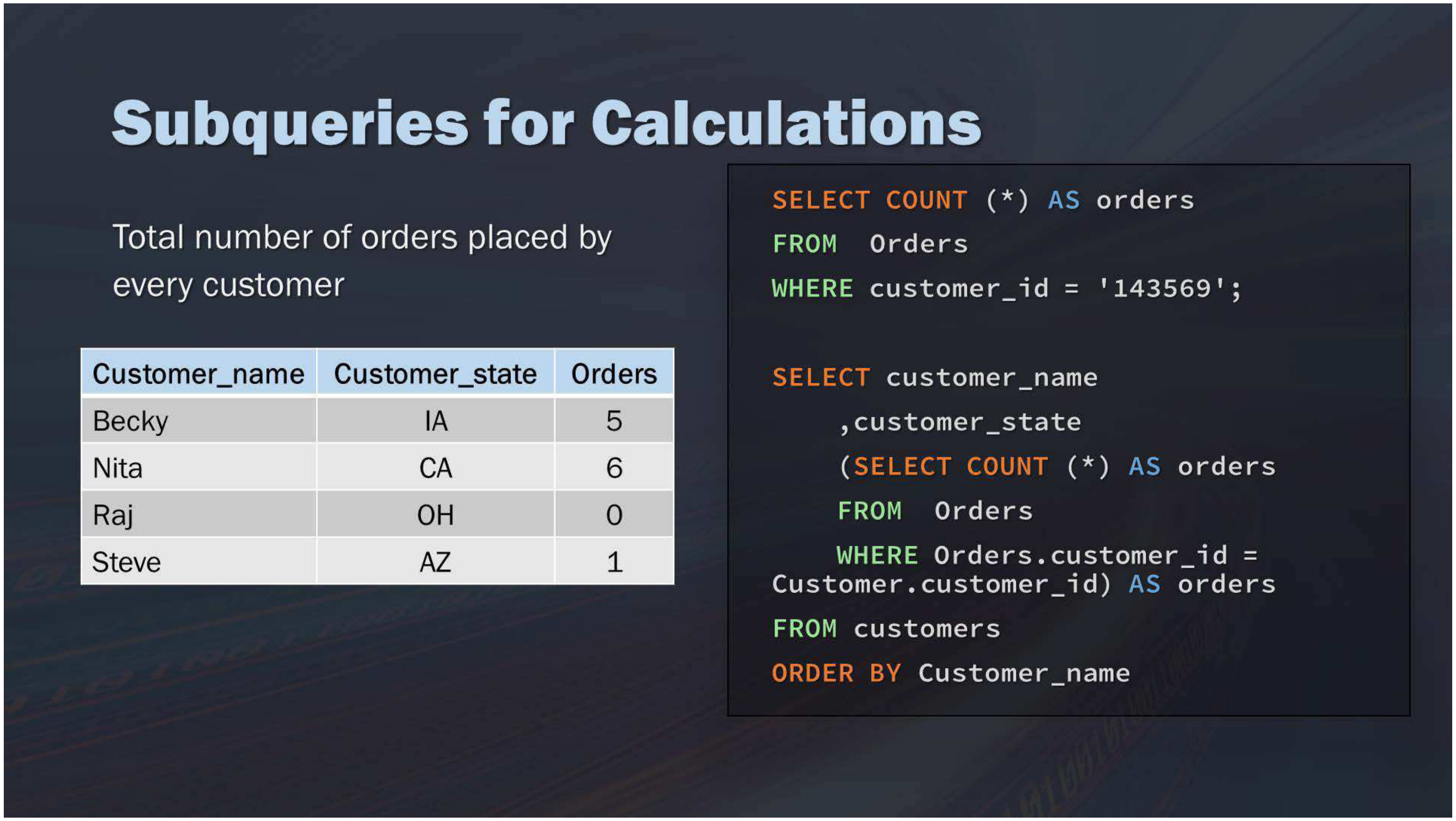
www.poorsql.com

Website will pre-format code

Uses proper indenting

Code is easier to read and troubleshoot

Slide 8: Subqueries for Calculations



Subqueries for Calculations

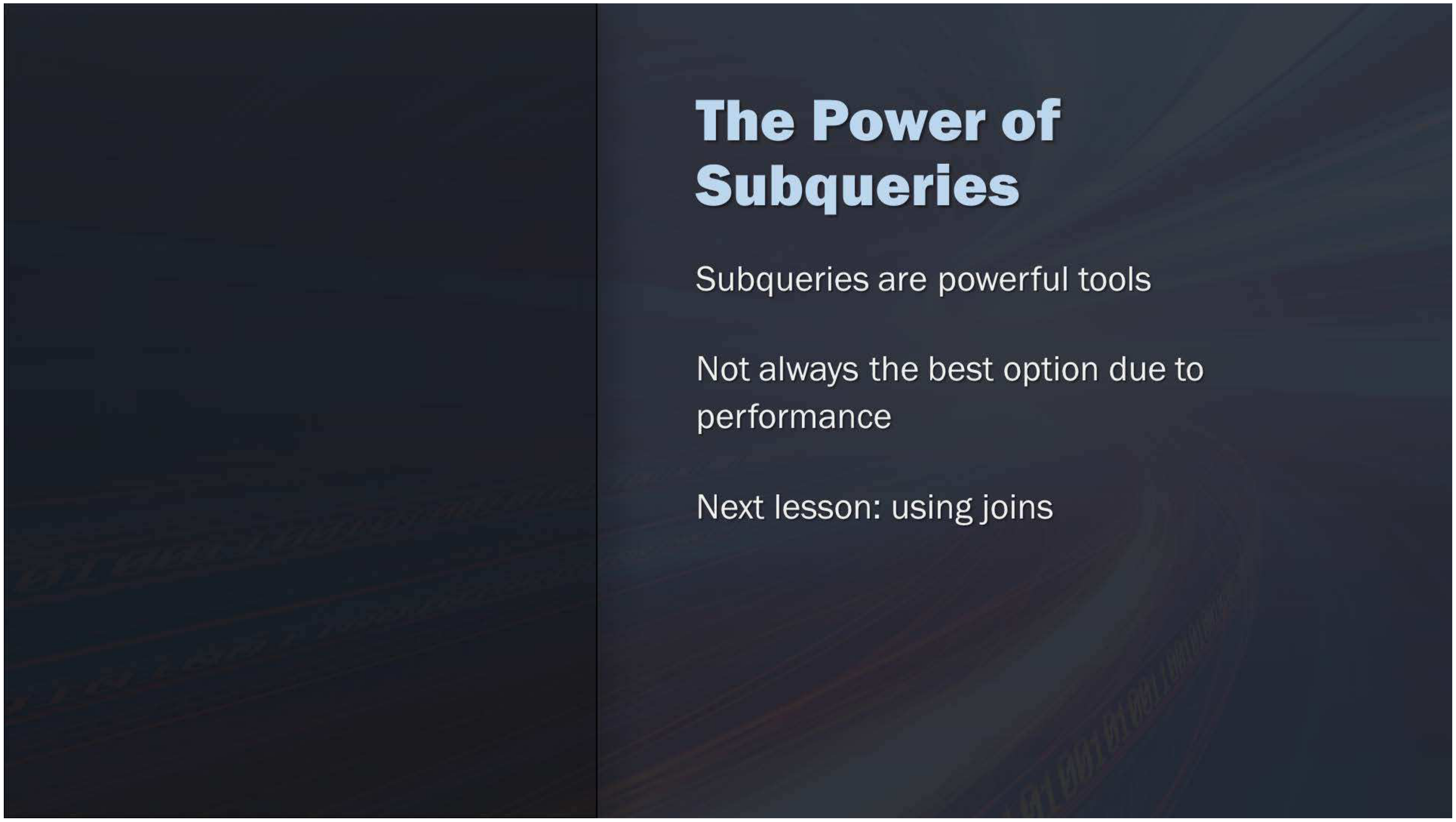
Total number of orders placed by every customer

Customer_name	Customer_state	Orders
Becky	IA	5
Nita	CA	6
Raj	OH	0
Steve	AZ	1

```
SELECT COUNT (*) AS orders
FROM  Orders
WHERE customer_id = '143569';

SELECT customer_name
    ,customer_state
    (SELECT COUNT (*) AS orders
    FROM  Orders
    WHERE Orders.customer_id =
Customer.customer_id) AS orders
FROM customers
ORDER BY Customer_name
```

Slide 9: The Power of Subqueries



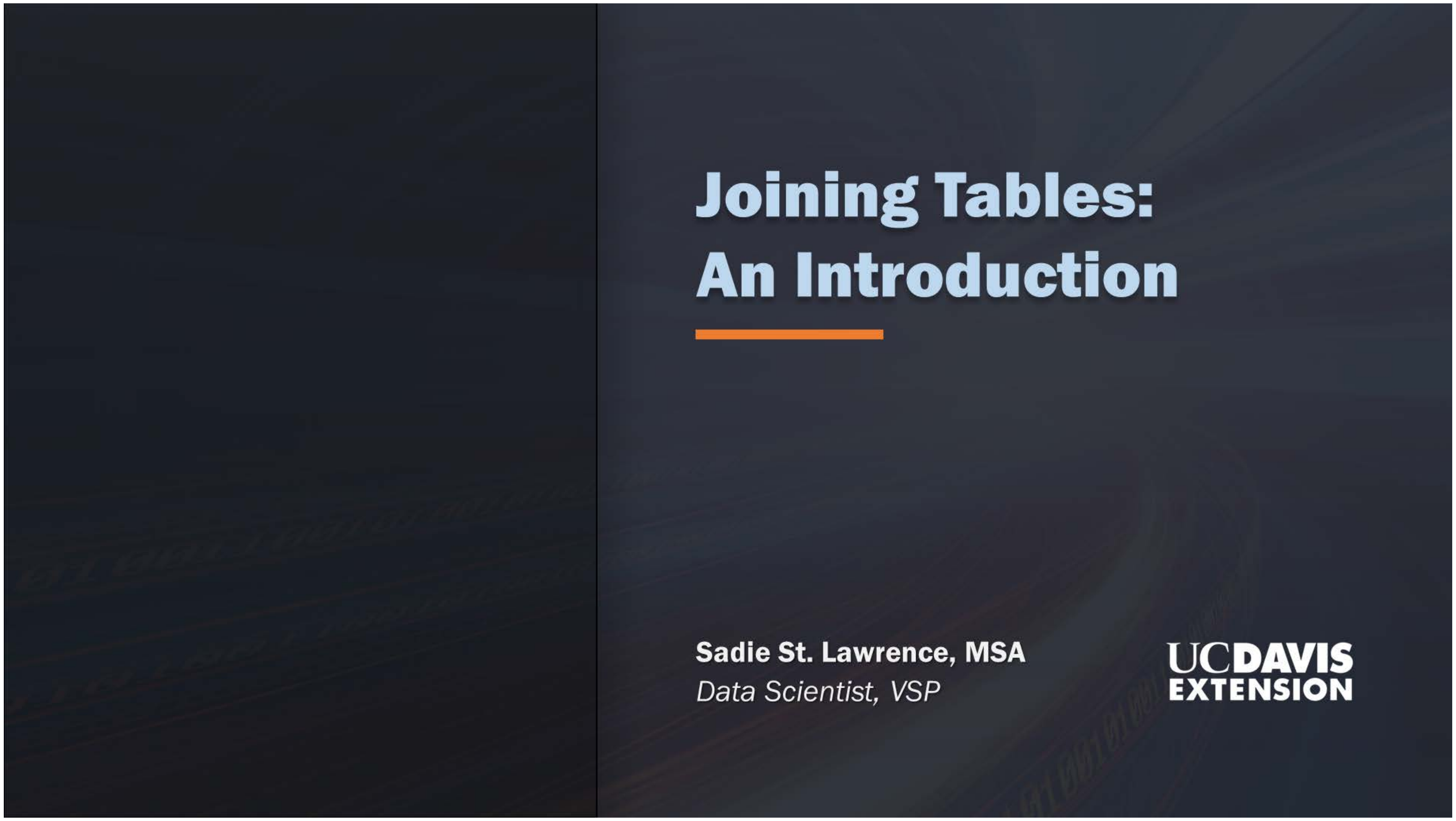
The Power of Subqueries

Subqueries are powerful tools

Not always the best option due to performance

Next lesson: using joins

Slide 1: Joining Tables: An Introduction

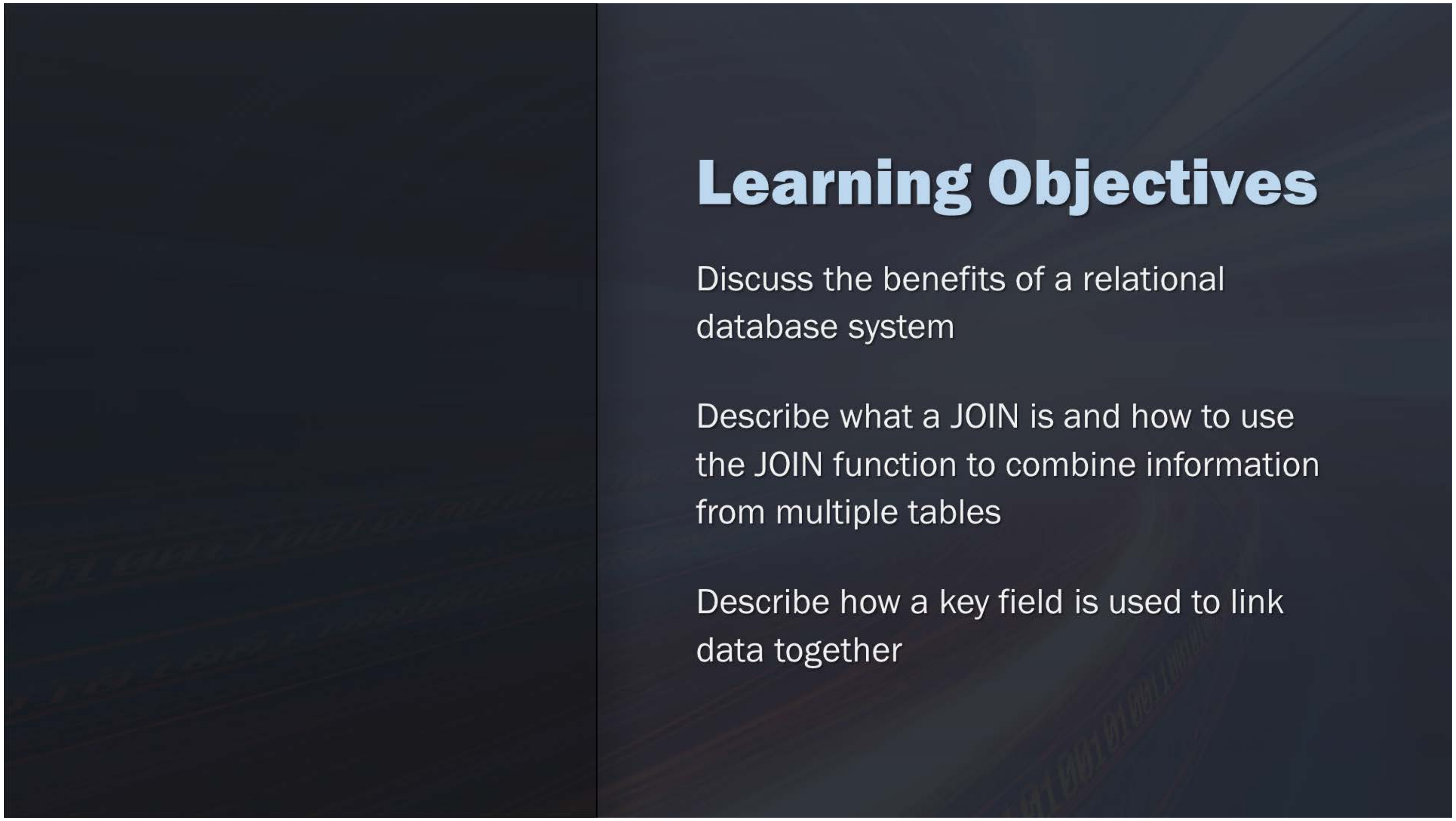


Joining Tables: An Introduction

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Learning Objectives



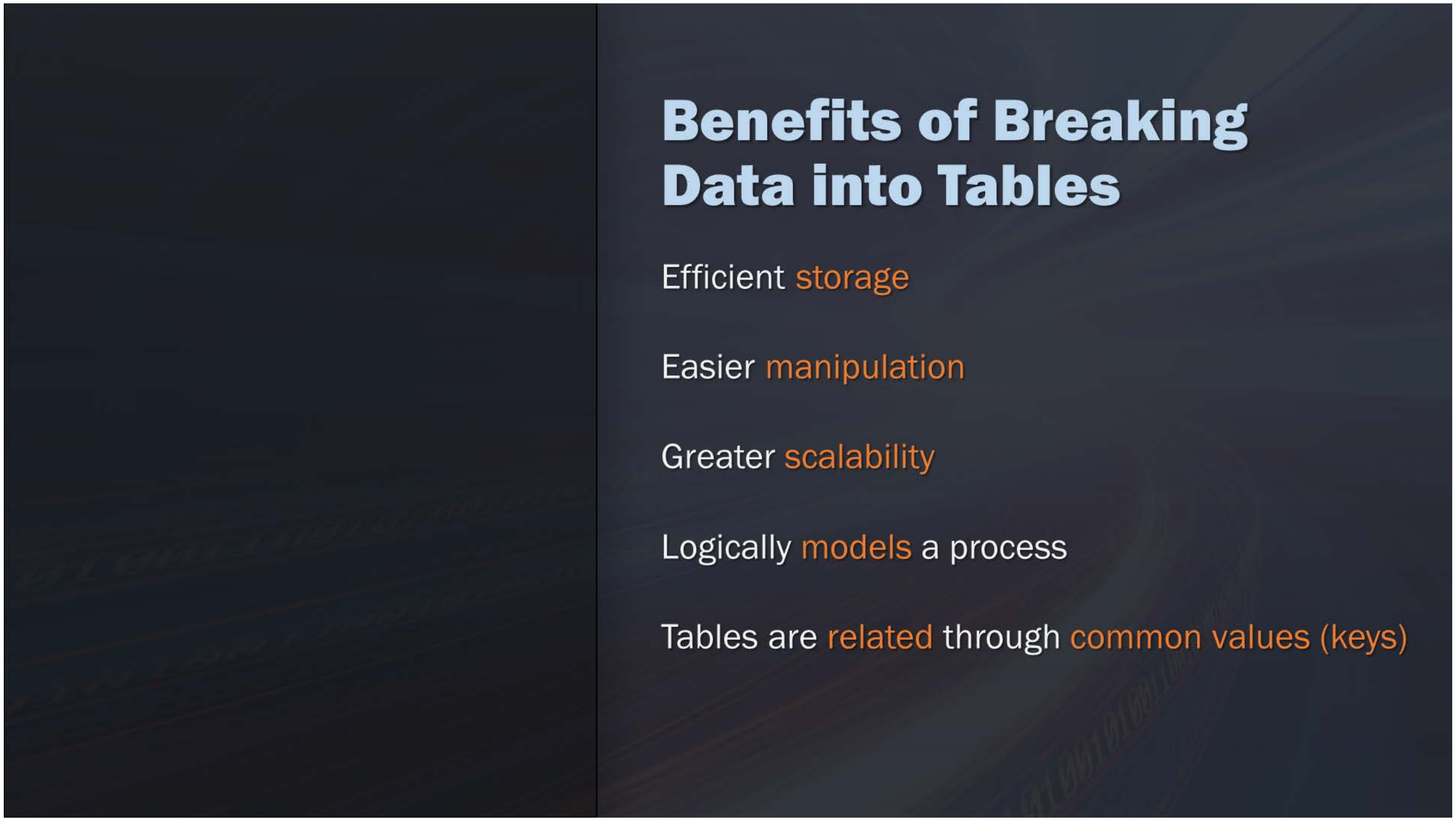
Learning Objectives

Discuss the benefits of a relational database system

Describe what a JOIN is and how to use the JOIN function to combine information from multiple tables

Describe how a key field is used to link data together

Slide 3: Benefits of Breaking Data into Tables



Benefits of Breaking Data into Tables

Efficient storage

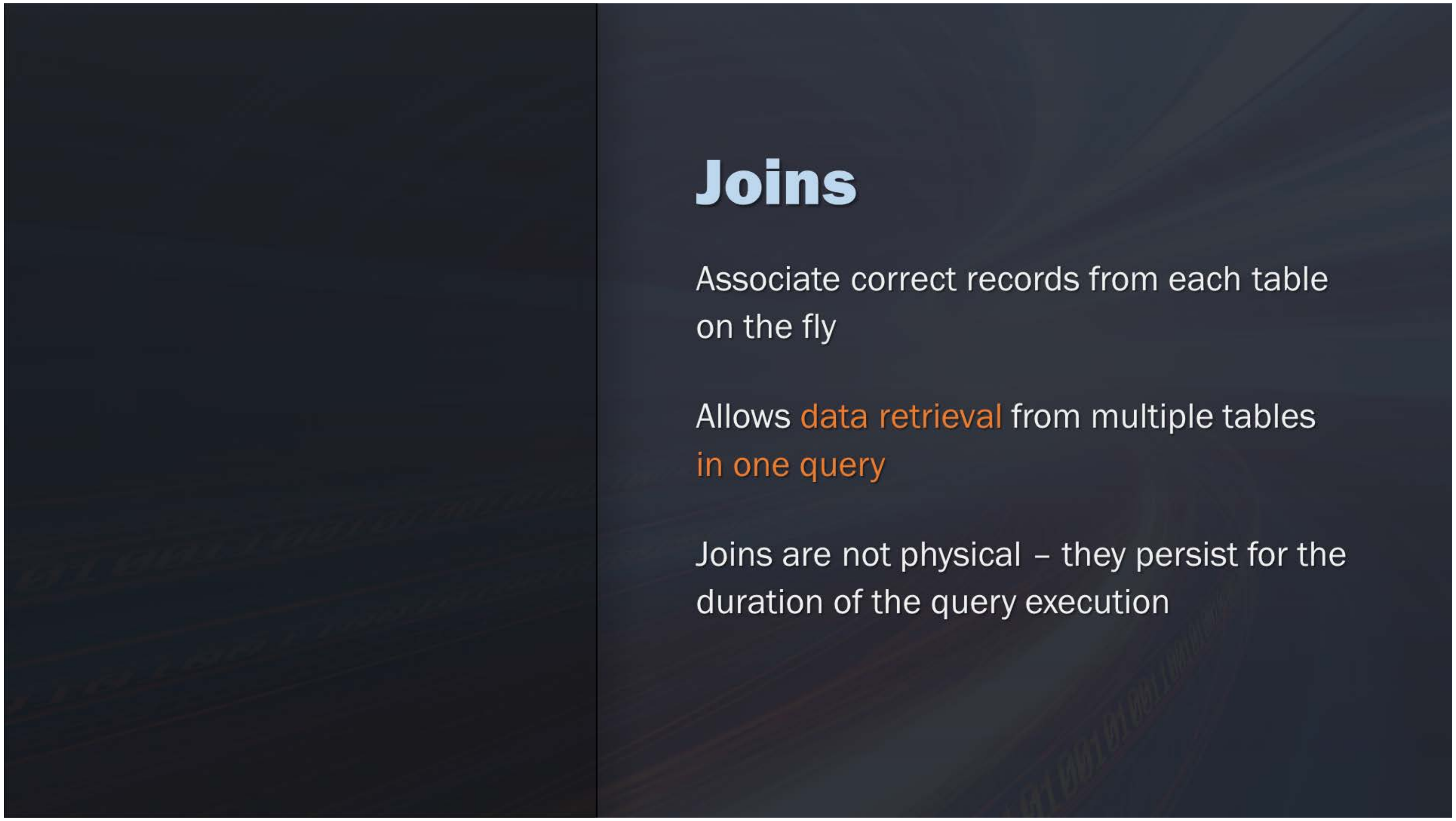
Easier manipulation

Greater scalability

Logically models a process

Tables are related through common values (keys)

Slide 4: Joins



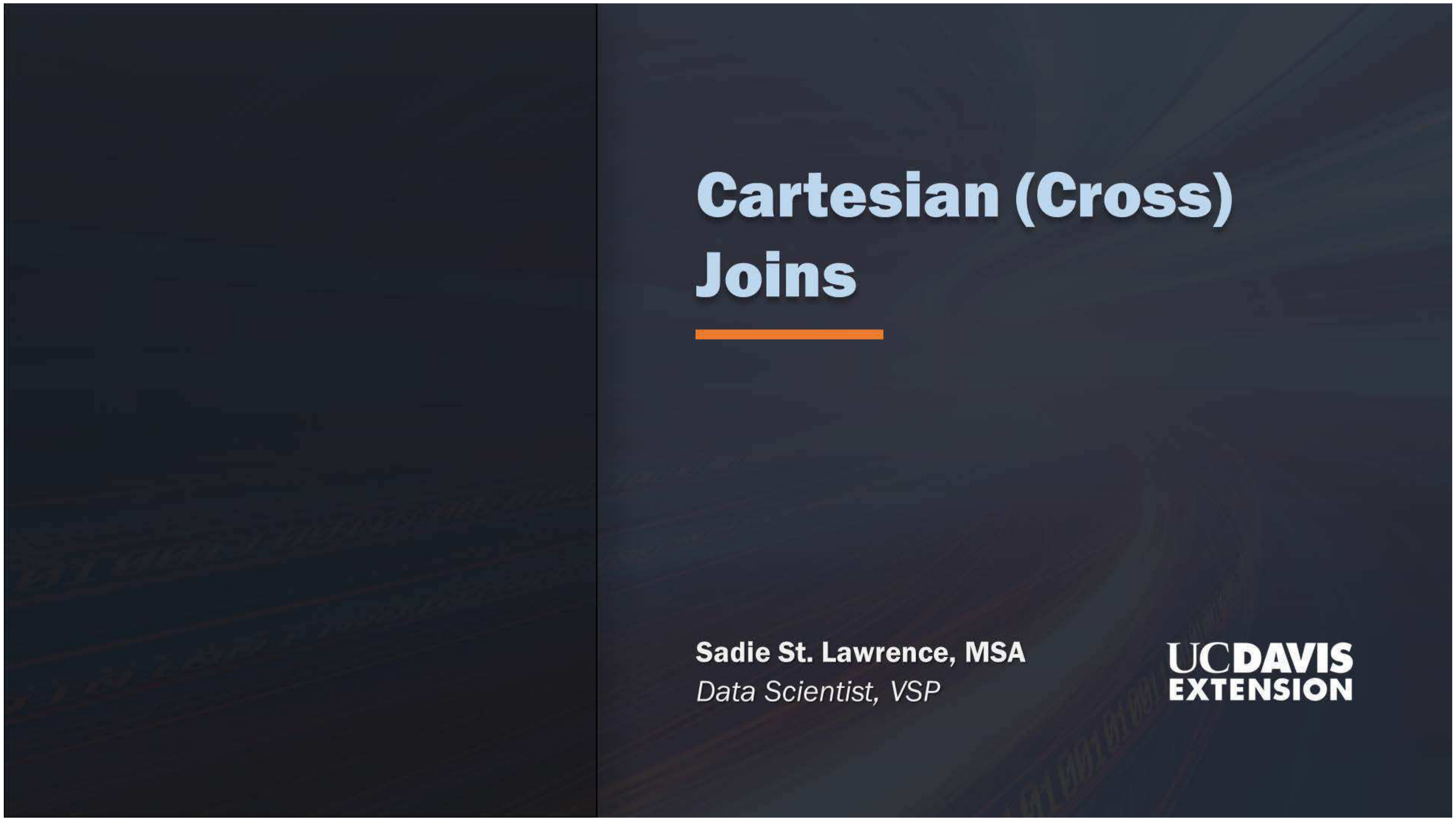
Joins

Associate correct records from each table on the fly

Allows data retrieval from multiple tables in one query

Joins are not physical – they persist for the duration of the query execution

Slide 1: Cartesian (Cross) Joins

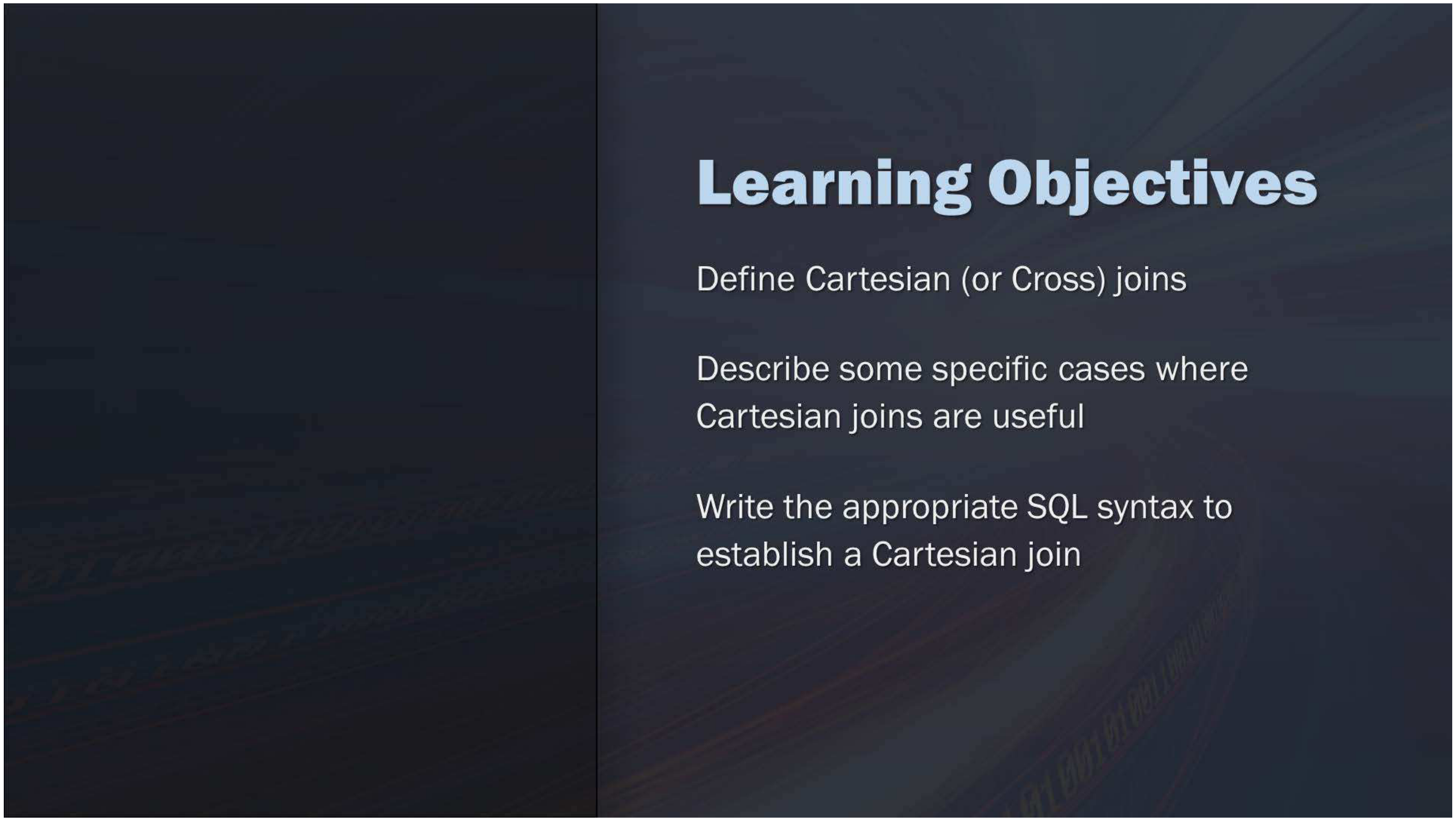


Cartesian (Cross) Joins

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Learning Objectives



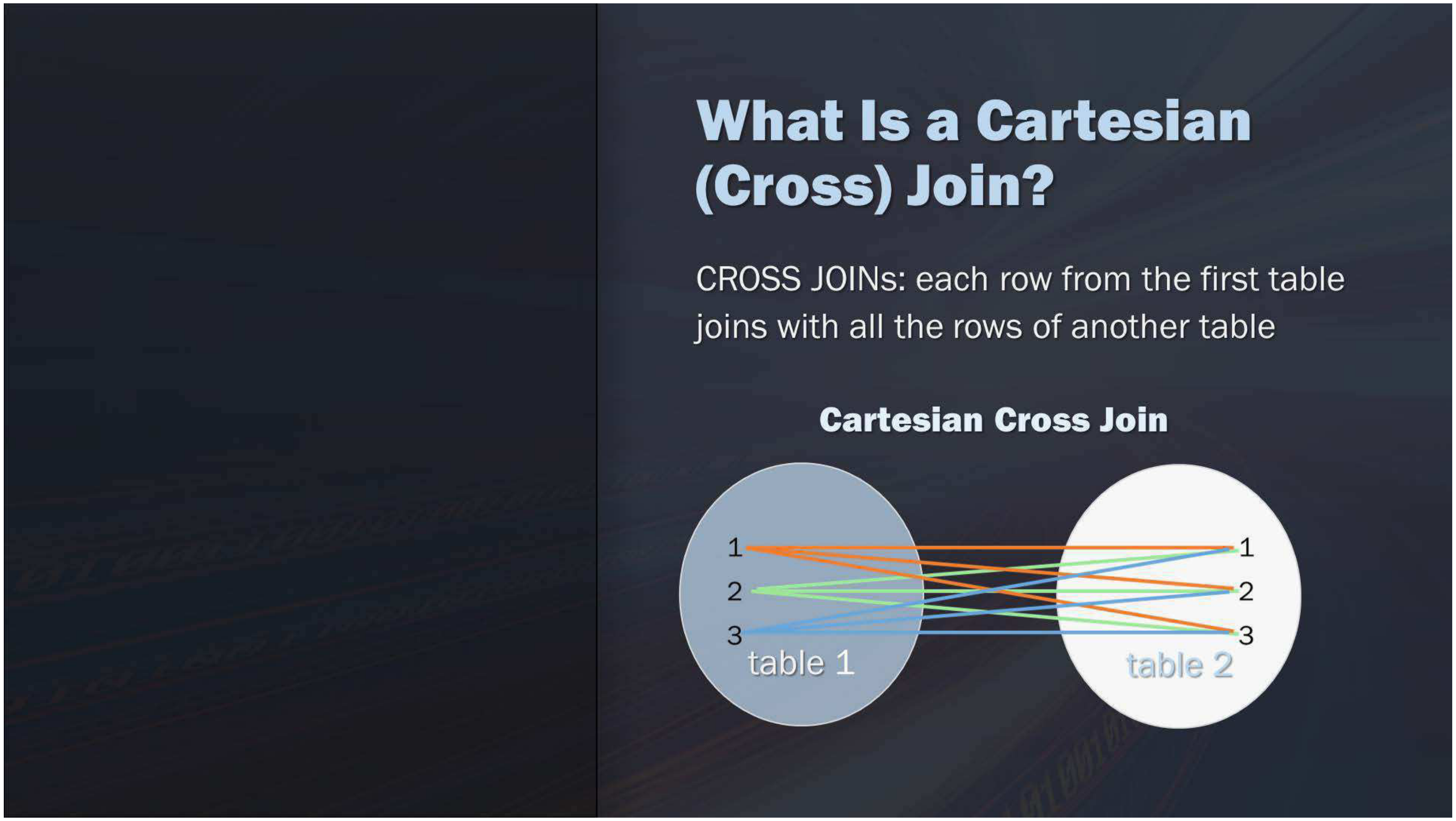
Learning Objectives

Define Cartesian (or Cross) joins

Describe some specific cases where Cartesian joins are useful

Write the appropriate SQL syntax to establish a Cartesian join

Slide 3: What Is a Cartesian (Cross) Join

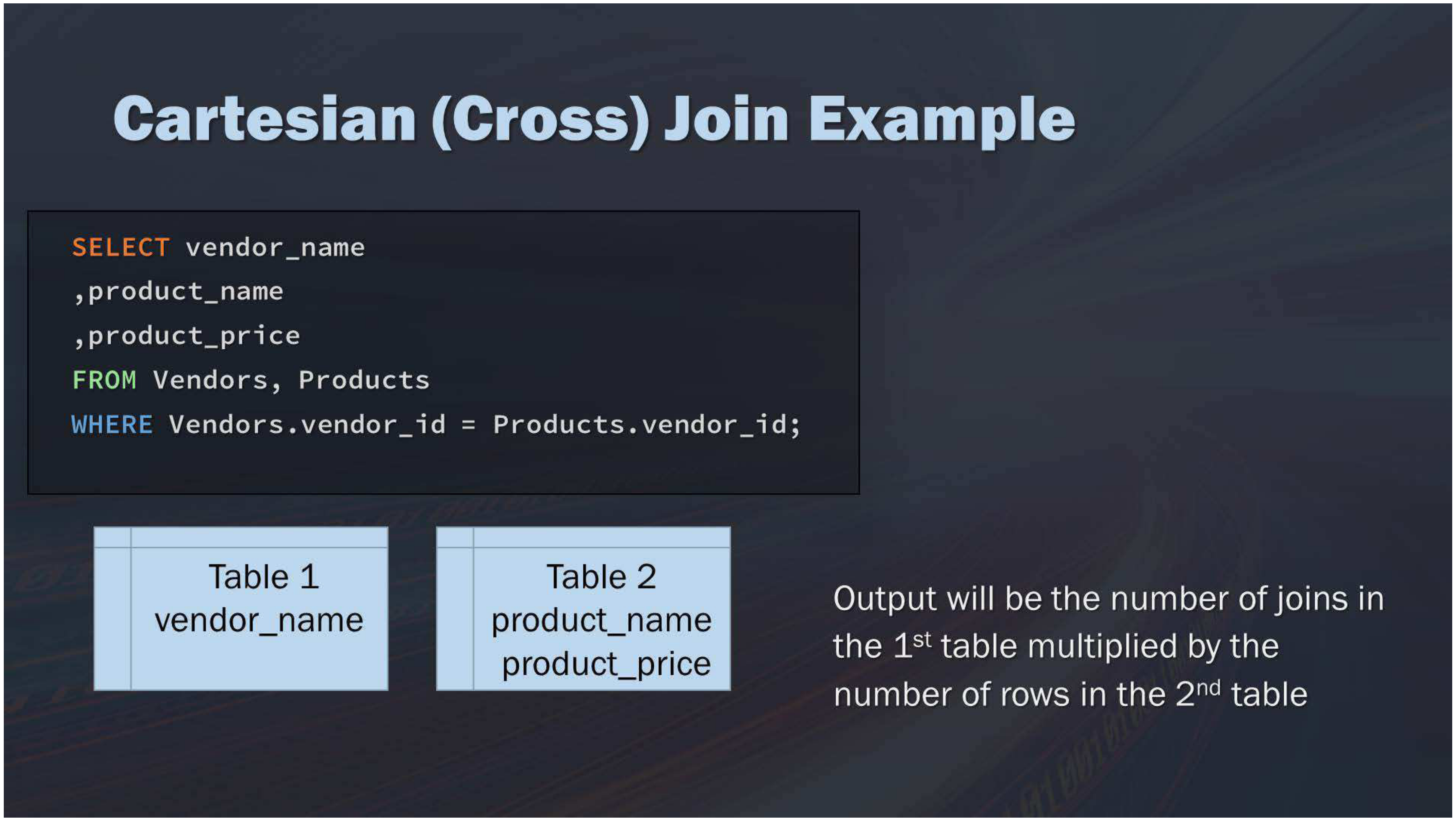


What Is a Cartesian (Cross) Join?

CROSS JOINs: each row from the first table joins with all the rows of another table

Cartesian Cross Join

Slide 4: Cartesian (Cross) Join Example



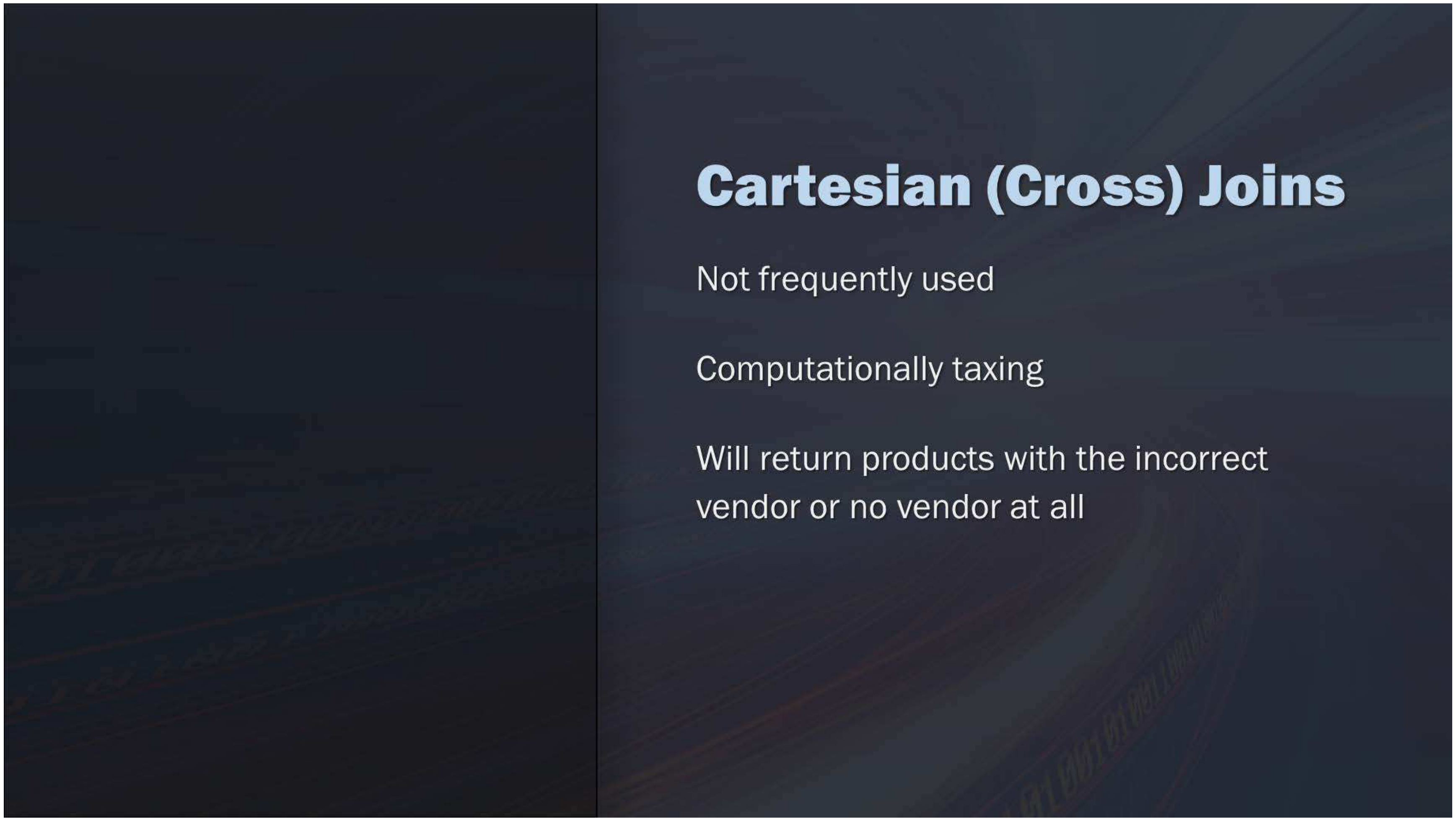
Cartesian (Cross) Join Example

```sql
SELECT vendor_name
,product_name
,product_price
FROM Vendors, Products
WHERE Vendors.vendor_id = Products.vendor_id;
```

Table 1
vendor_name
Table 2
product_name
product_price

Output will be the number of joins in the 1st table multiplied by the number of rows in the 2nd table

Slide 5: Cartesian (Cross) Joins



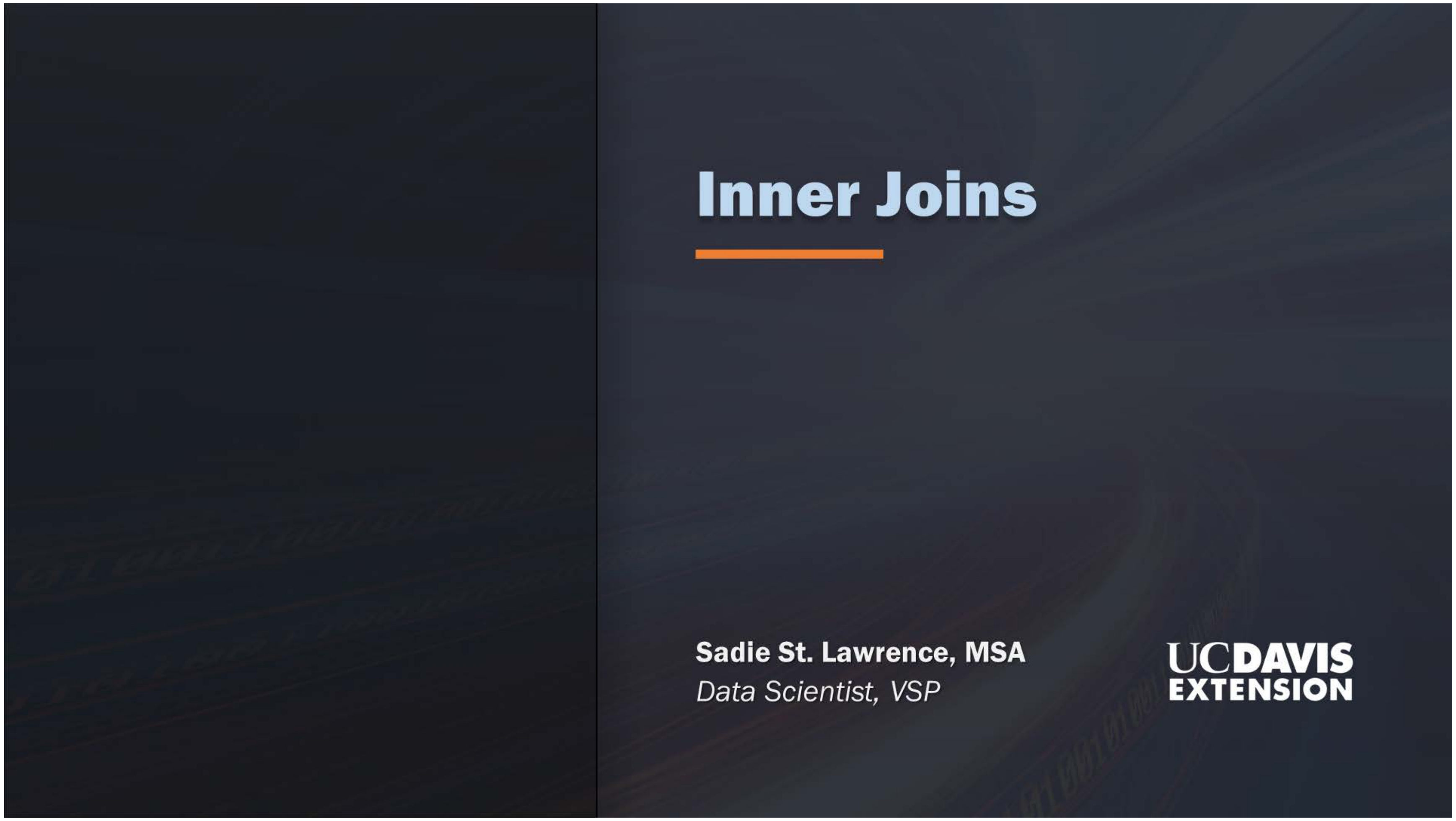
Cartesian (Cross) Joins

Not frequently used

Computationally taxing

Will return products with the incorrect vendor or no vendor at all

Slide 1: Inner Joins

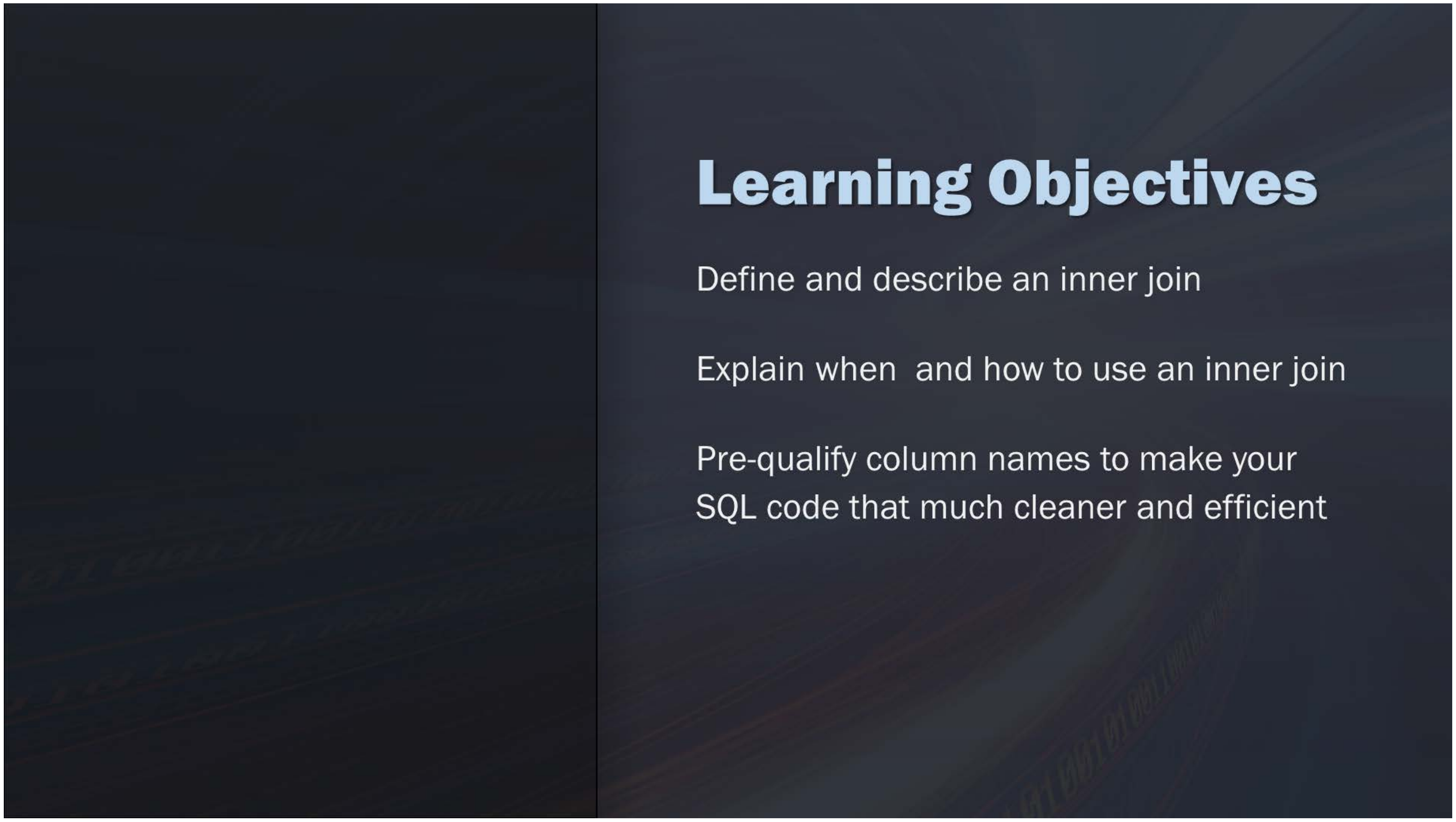


Inner Joins

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Learning Objectives



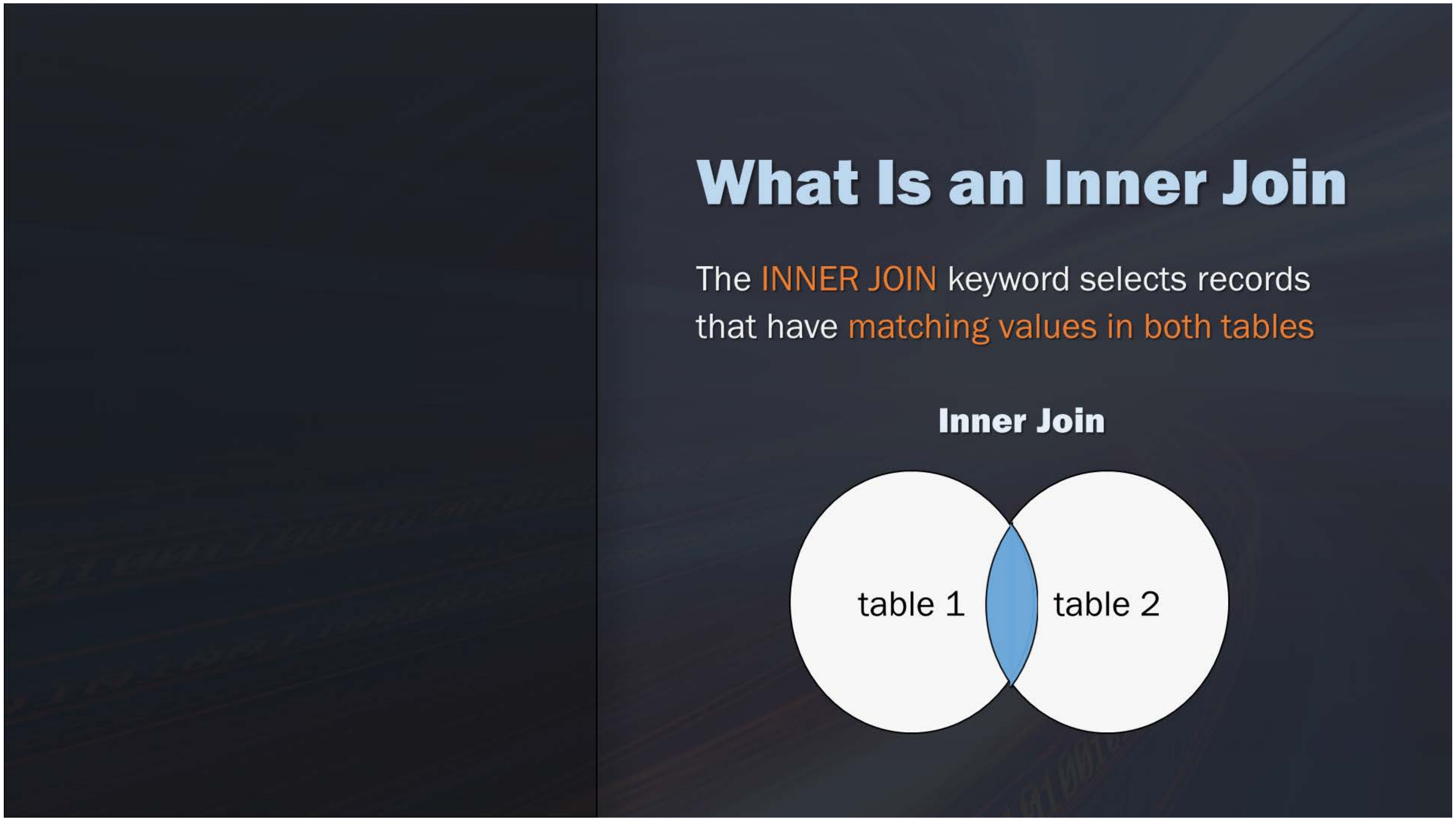
Learning Objectives

Define and describe an inner join

Explain when and how to use an inner join

Pre-qualify column names to make your SQL code that much cleaner and efficient

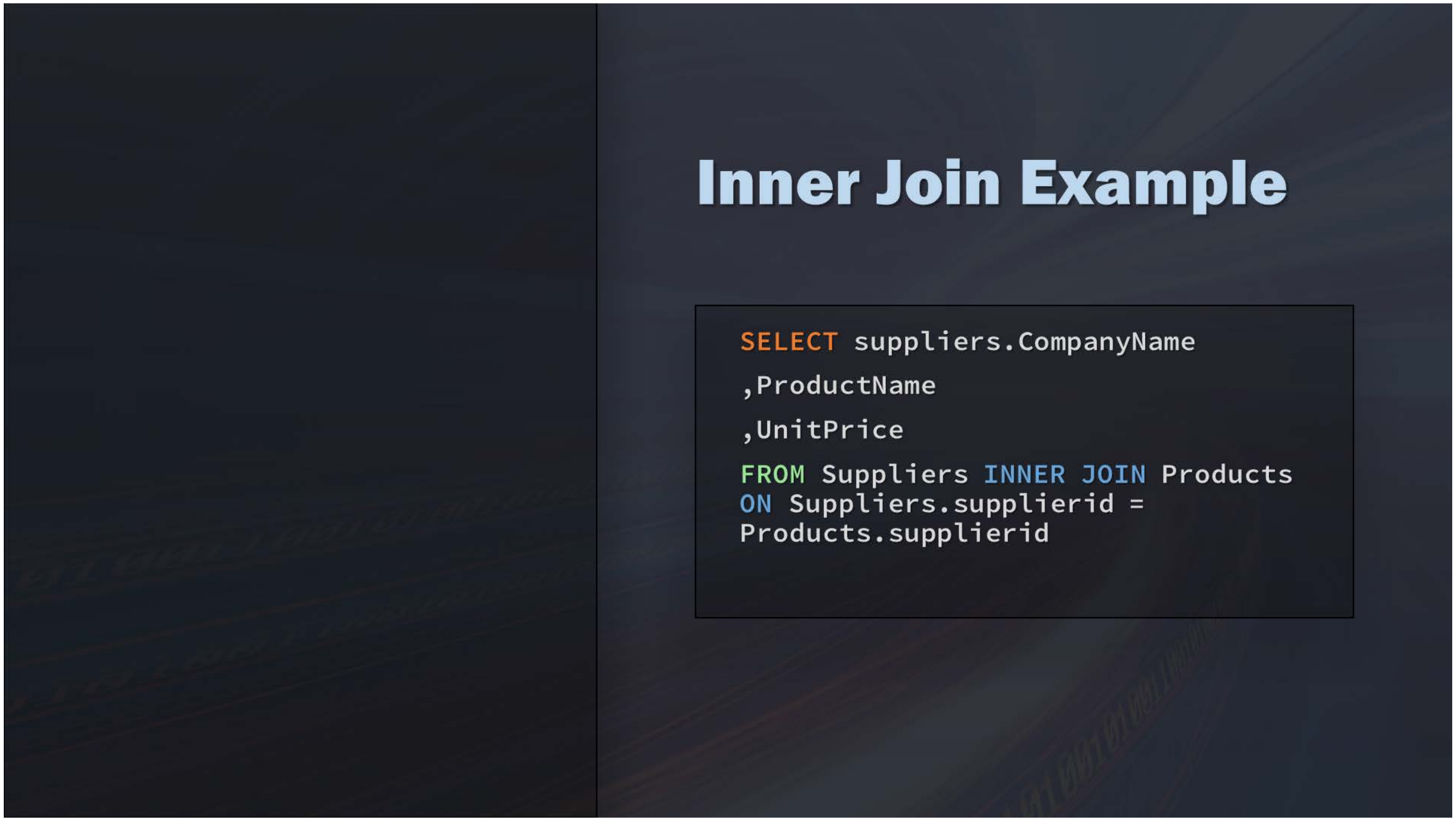
Slide 3: What Is an Inner Join



What Is an Inner Join

The INNER JOIN keyword selects records that have matching values in both tables

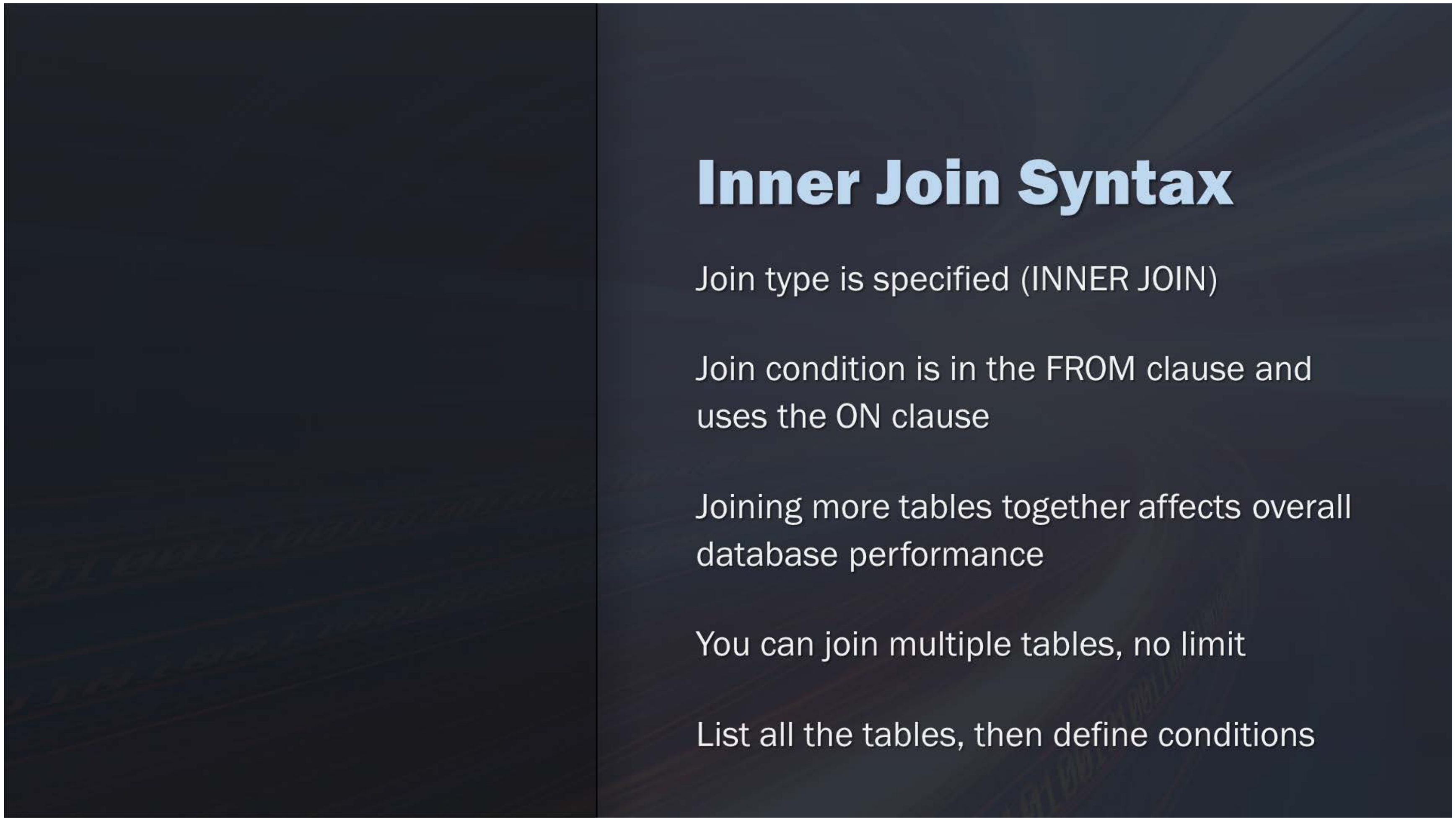
Slide 4: Inner Join Example



Inner Join Example

```
SELECT suppliers.CompanyName
,ProductName
,UnitPrice
FROM Suppliers INNER JOIN Products
ON Suppliers.supplierid =
Products.supplierid
```

Slide 5: Inner Join Syntax



Inner Join Syntax

Join type is specified (INNER JOIN)

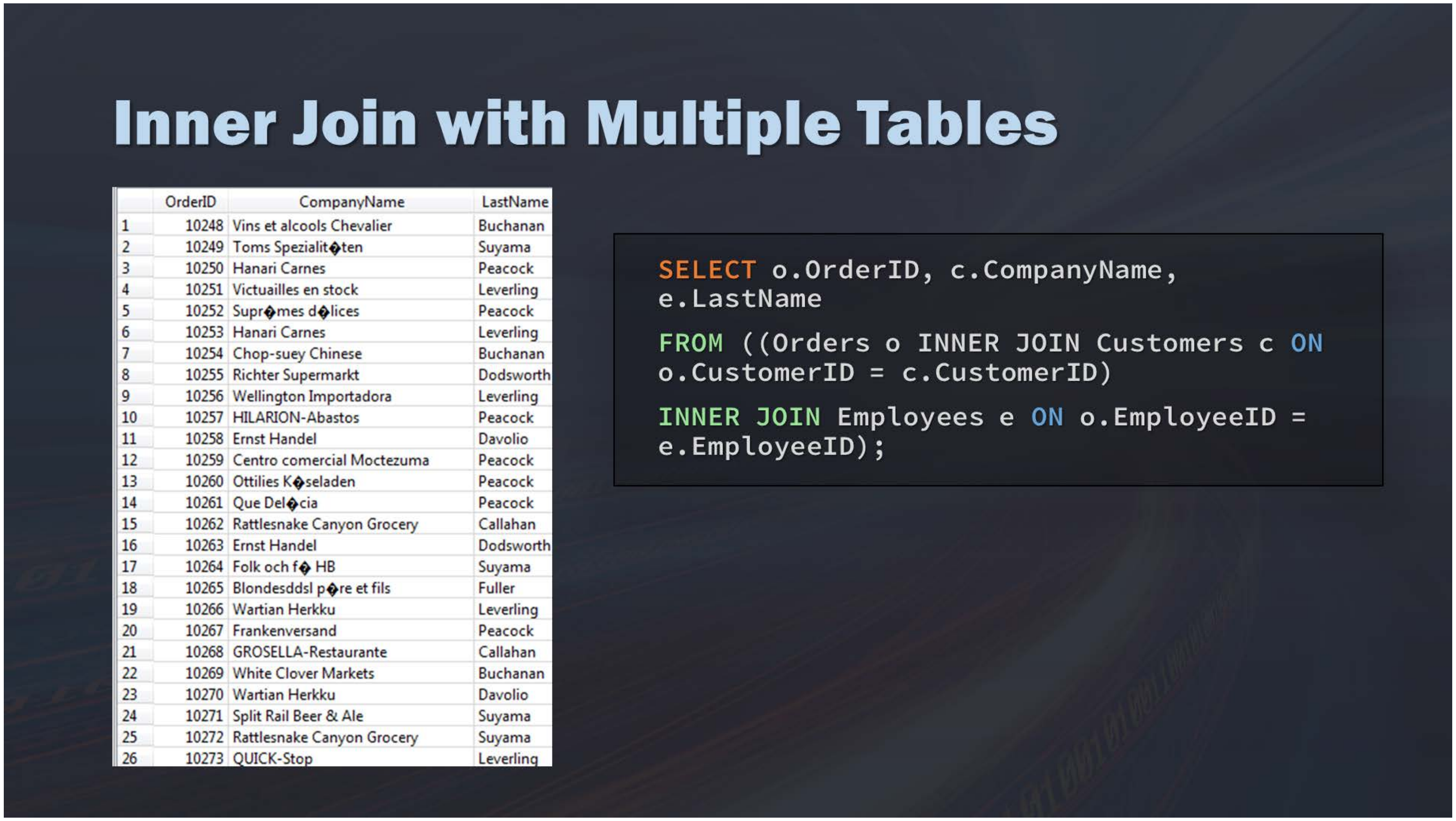
Join condition is in the FROM clause and uses the ON clause

Joining more tables together affects overall database performance

You can join multiple tables, no limit

List all the tables, then define conditions

Slide 6: Inner Join with Multiple Tables

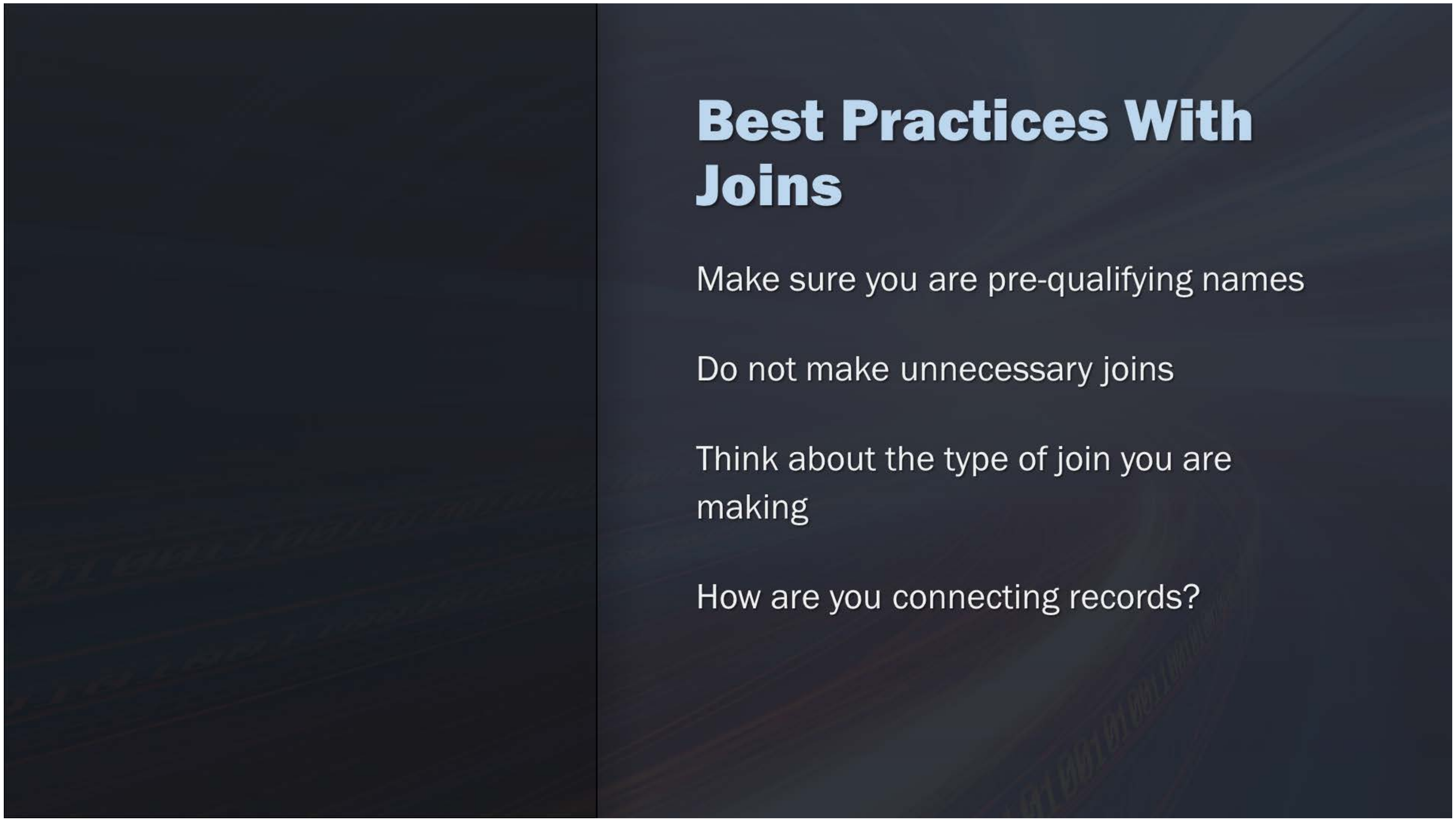


Inner Join with Multiple Tables

	OrderID	CompanyName	LastName
1	10248	Vins et alcools Chevalier	Buchanan
2	10249	Toms Spezialit�ten	Suyama
3	10250	Hanari Carnes	Peacock
4	10251	Victuailles en stock	Leverling
5	10252	Supr�mes d�lices	Peacock
6	10253	Hanari Carnes	Leverling
7	10254	Chop-suey Chinese	Buchanan
8	10255	Richter Supermarkt	Dodsworth
9	10256	Wellington Importadora	Leverling
10	10257	HILARION-Abastos	Peacock
11	10258	Ernst Handel	Davolio
12	10259	Centro comercial Moctezuma	Peacock
13	10260	Ottilies K�seladen	Peacock
14	10261	Que Del�cia	Peacock
15	10262	Rattlesnake Canyon Grocery	Callahan
16	10263	Ernst Handel	Dodsworth
17	10264	Folk och f� HB	Suyama
18	10265	Blondesddsl p�re et fils	Fuller
19	10266	Wartian Herkku	Leverling
20	10267	Frankenversand	Peacock
21	10268	GROSELLA-Restaurante	Callahan
22	10269	White Clover Markets	Buchanan
23	10270	Wartian Herkku	Davolio
24	10271	Split Rail Beer & Ale	Suyama
25	10272	Rattlesnake Canyon Grocery	Suyama
26	10273	QUICK-Stop	Leverling

```
SELECT o.OrderID, c.CompanyName,
e.LastName
FROM ((Orders o INNER JOIN Customers c ON
o.CustomerID = c.CustomerID)
INNER JOIN Employees e ON o.EmployeeID =
e.EmployeeID);
```

Slide 7: Best Practices with Joins



Best Practices With Joins

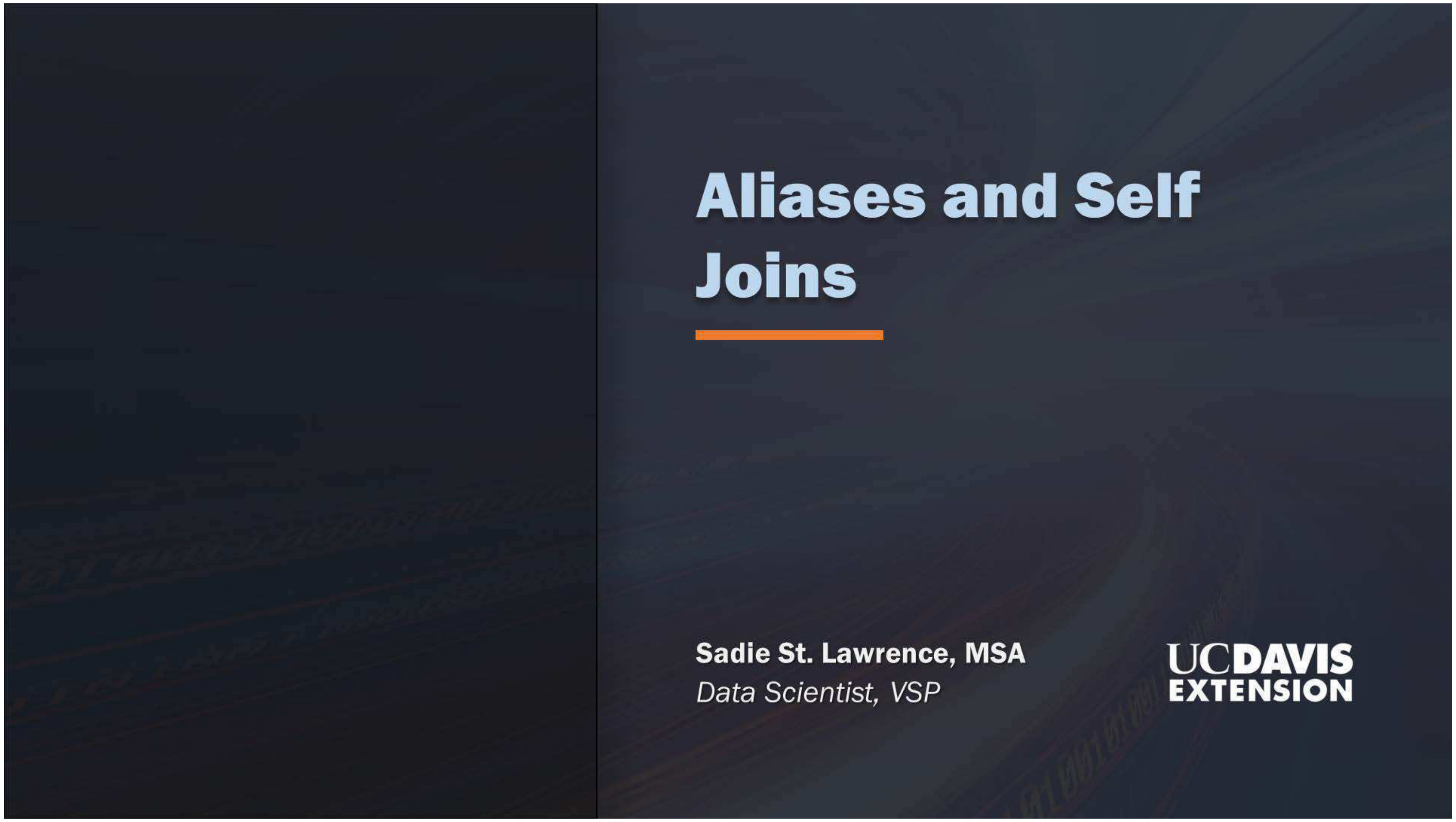
Make sure you are pre-qualifying names

Do not make unnecessary joins

Think about the type of join you are making

How are you connecting records?

Slide 1: Aliases and Self Joins

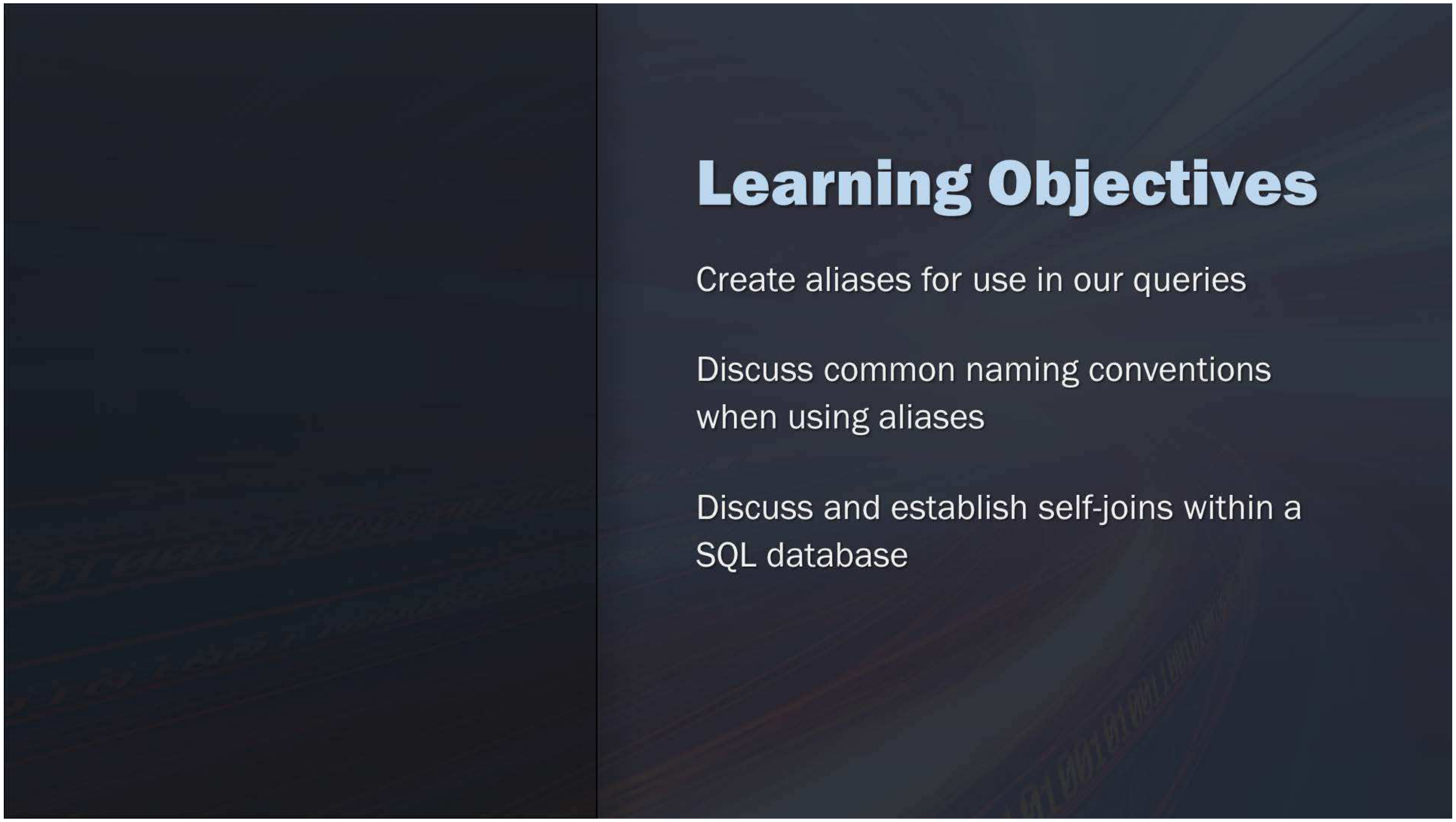


Aliases and Self Joins

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Learning Objectives



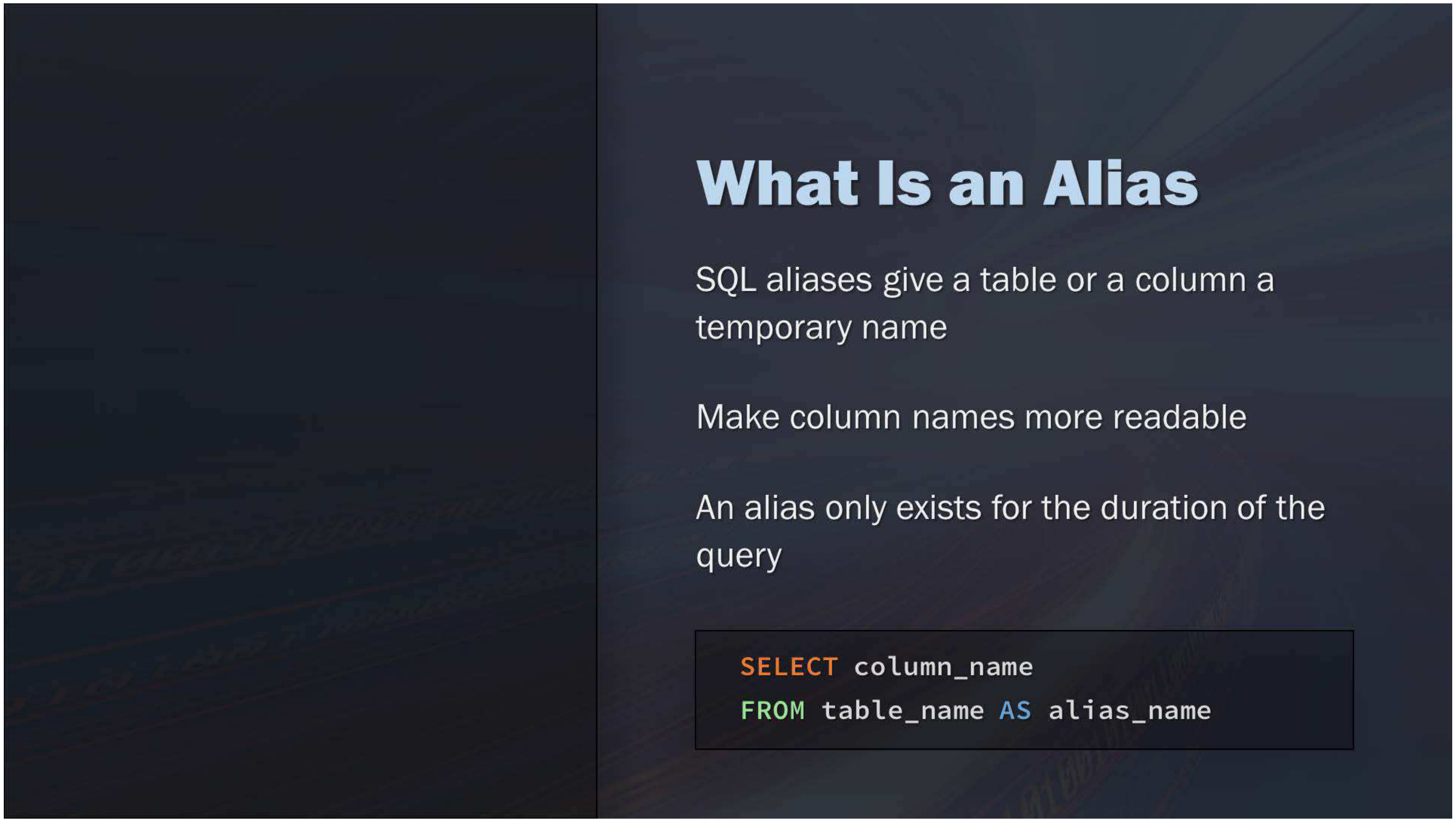
Learning Objectives

Create aliases for use in our queries

Discuss common naming conventions when using aliases

Discuss and establish self-joins within a SQL database

Slide 3: What Is an Alias



What Is an Alias

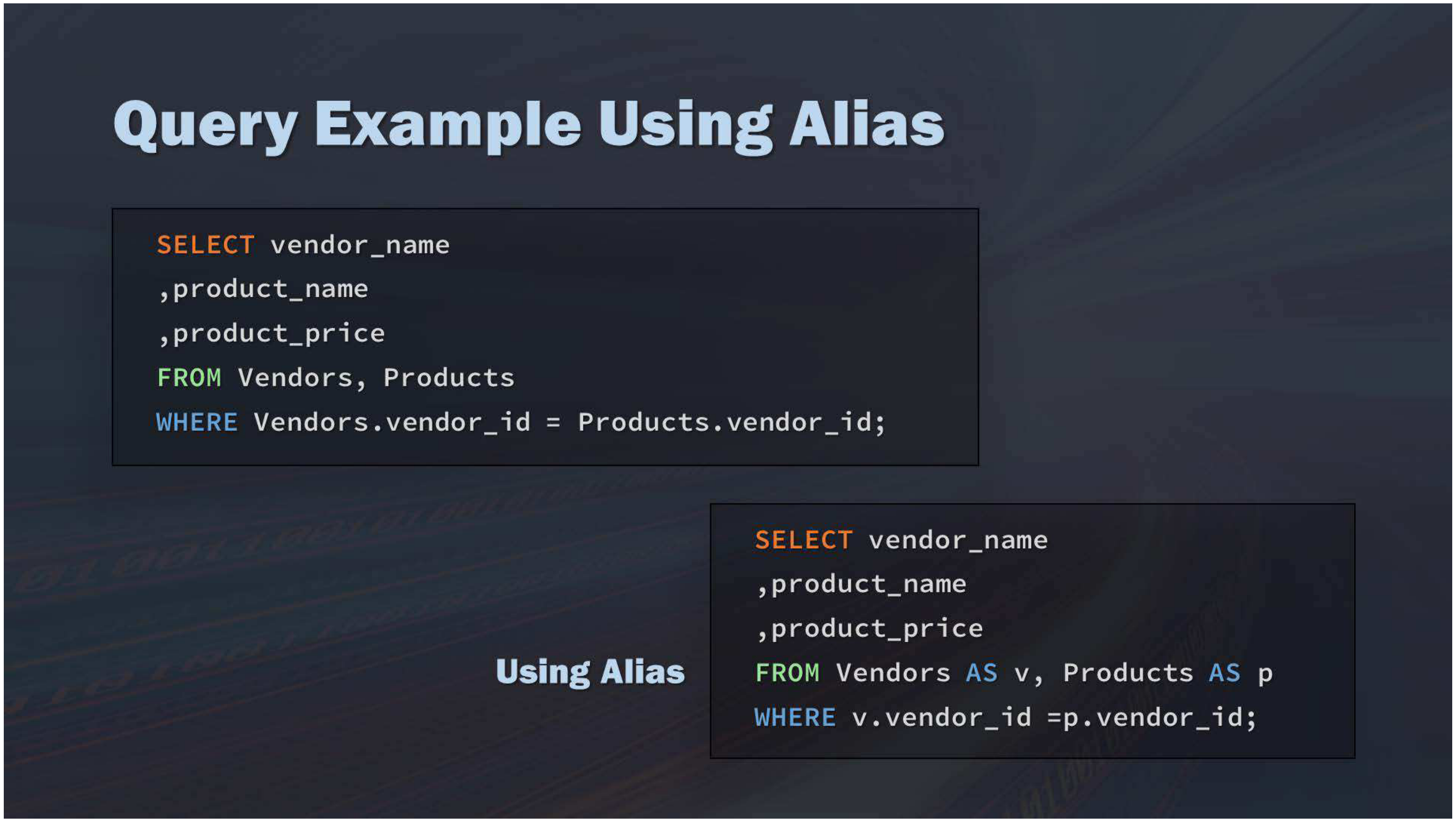
SQL aliases give a table or a column a temporary name

Make column names more readable

An alias only exists for the duration of the query

```
SELECT column_name
FROM table_name AS alias_name
```

Slide 4: Query Example Using Alias



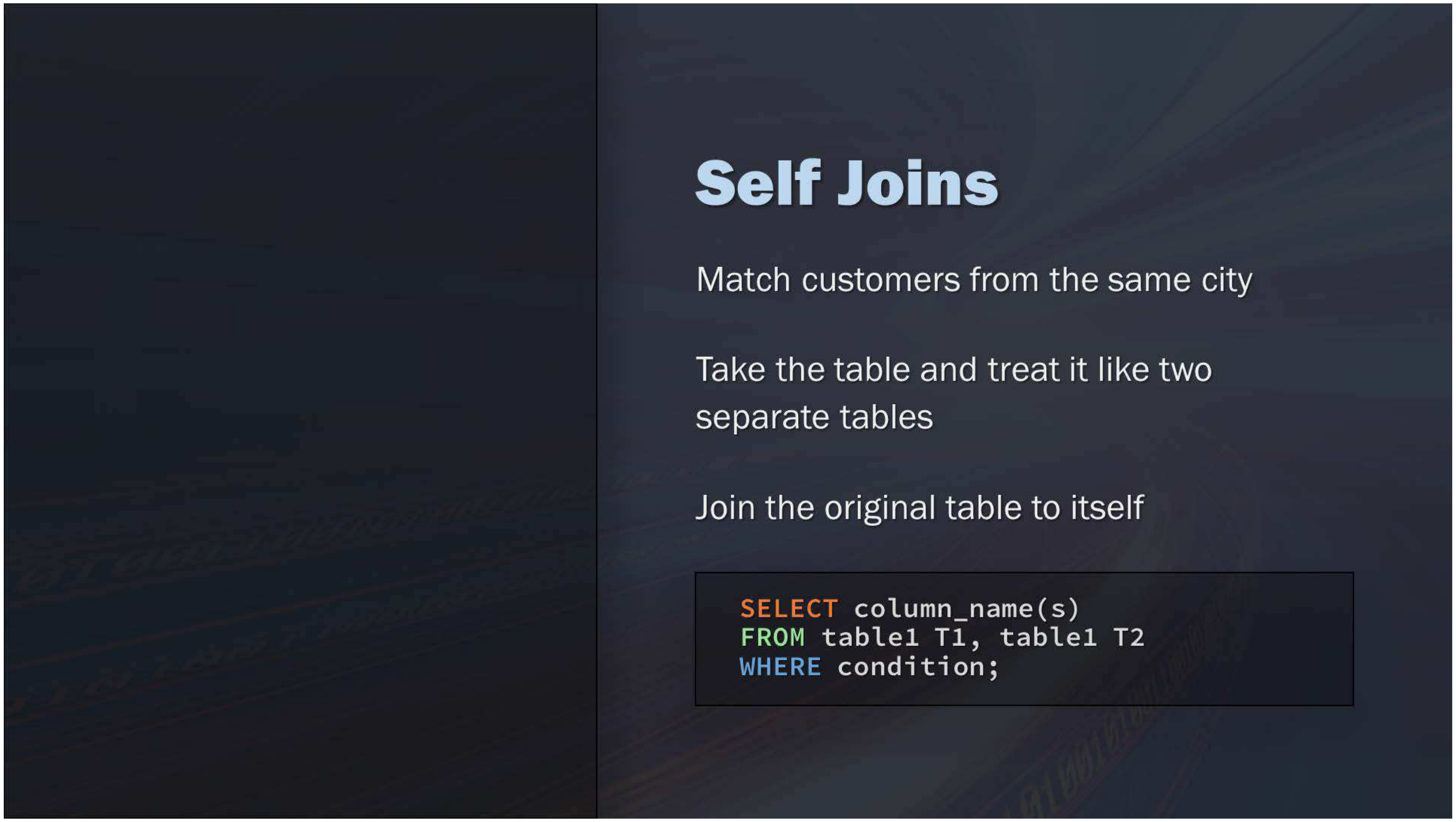
Query Example Using Alias

```
SELECT vendor_name
,product_name
,product_price
FROM Vendors, Products
WHERE Vendors.vendor_id = Products.vendor_id;
```

Using Alias

```
SELECT vendor_name
,product_name
,product_price
FROM Vendors AS v, Products AS p
WHERE v.vendor_id =p.vendor_id;
```

Slide 5: Self Joins



Self Joins

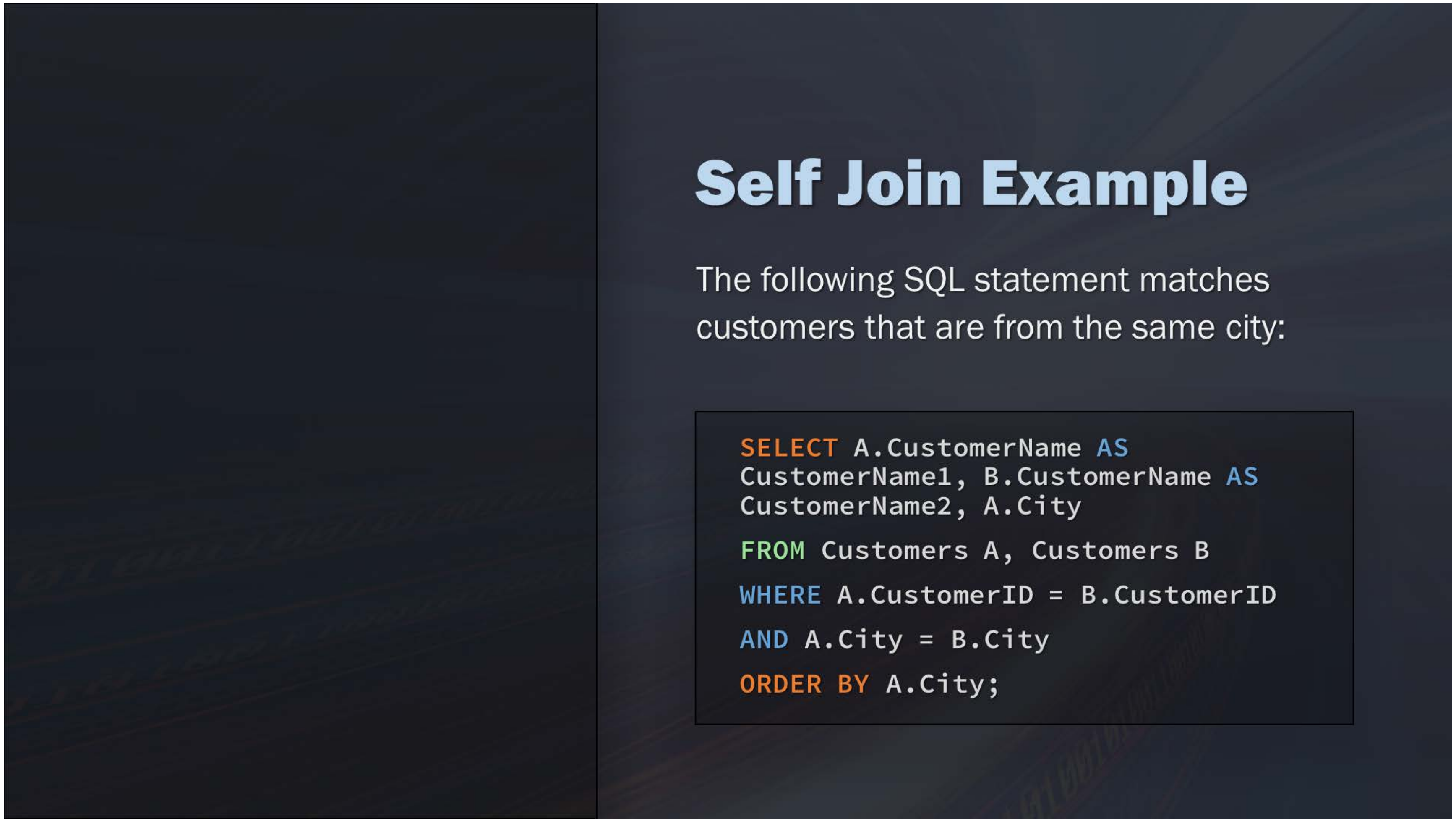
Match customers from the same city

Take the table and treat it like two separate tables

Join the original table to itself

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

Slide 6: Self Join Example



Self Join Example

The following SQL statement matches customers that are from the same city:

```
SELECT A.CustomerName AS
CustomerName1, B.CustomerName AS
CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID = B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

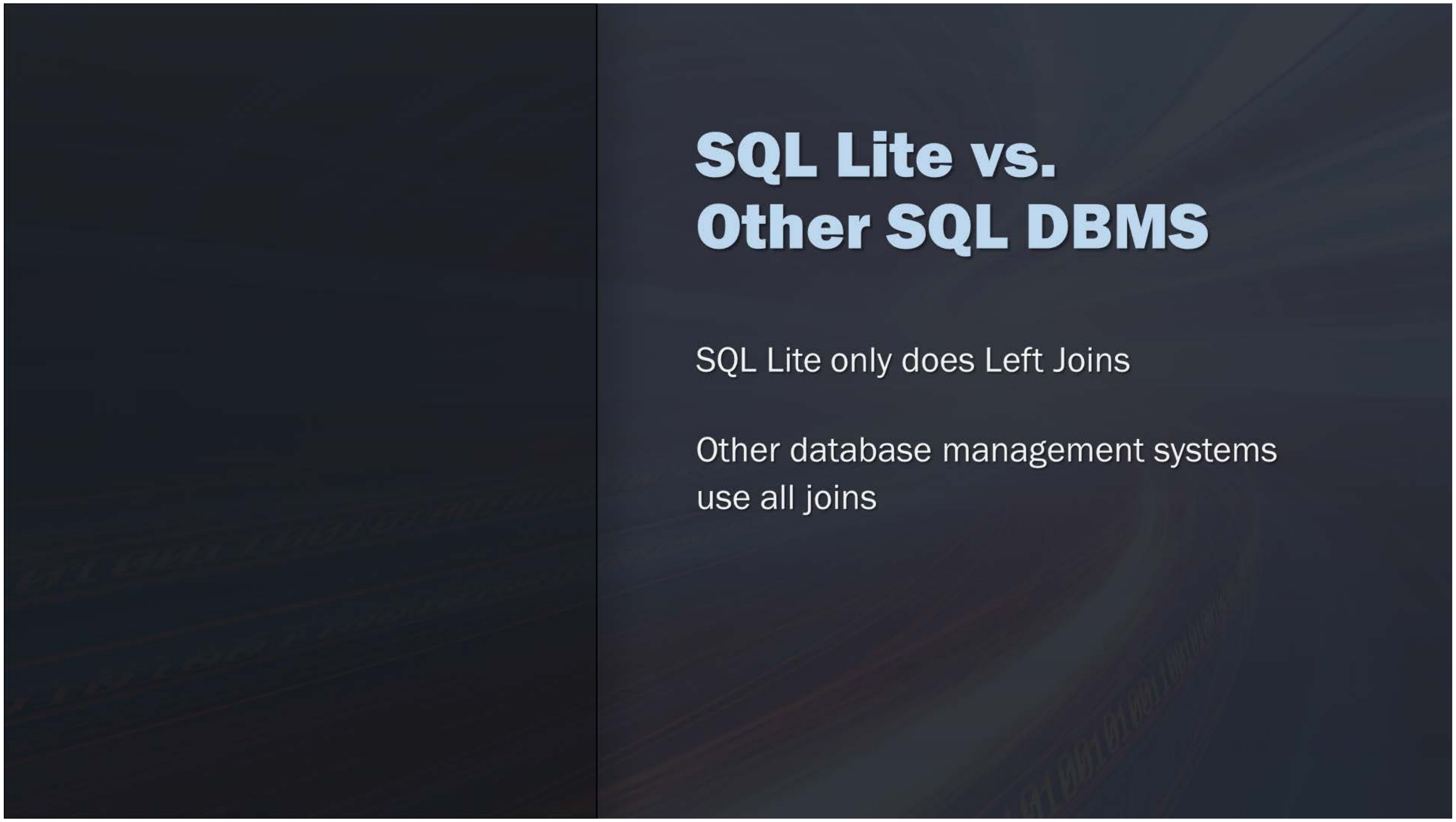
Slide 1: Advanced Joins: Left, Right and Full Outer Joins

Advanced Joins: Left, Right and Full Outer Joins

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: SQL Lite vs. Other SQL DBMS

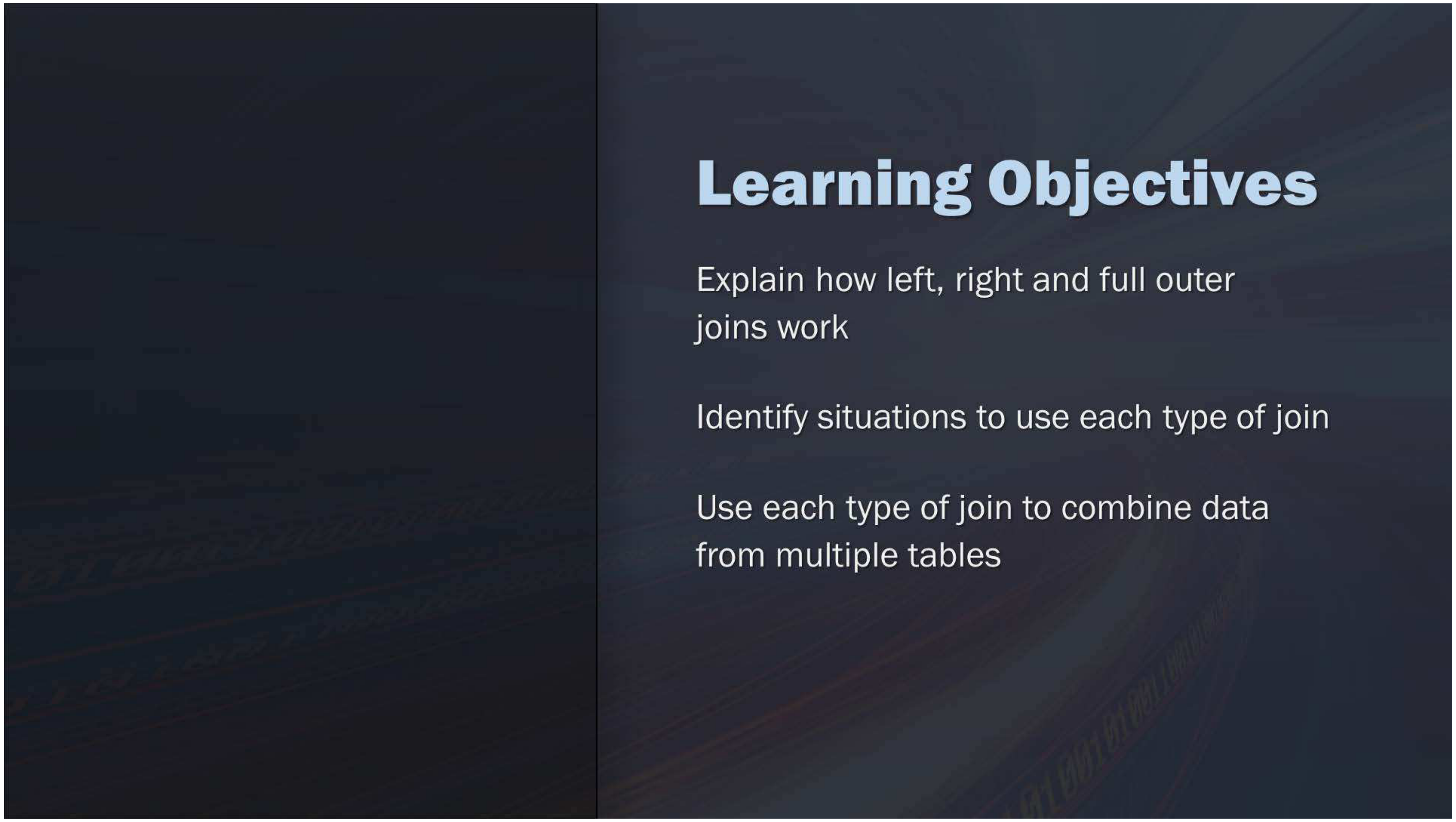


SQL Lite vs. Other SQL DBMS

SQL Lite only does Left Joins

Other database management systems use all joins

Slide 3: Learning Objectives



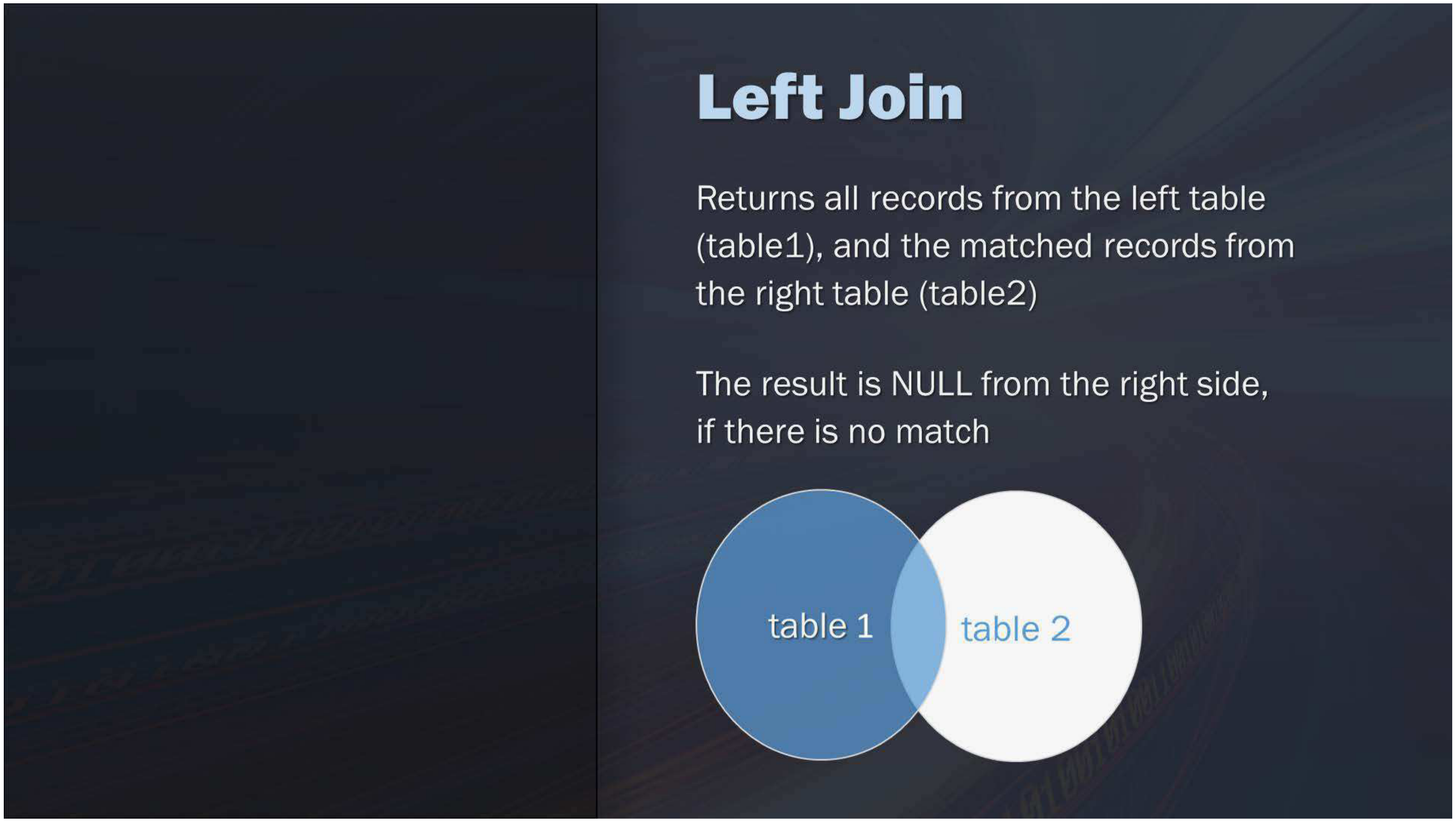
Learning Objectives

Explain how left, right and full outer joins work

Identify situations to use each type of join

Use each type of join to combine data from multiple tables

Slide 4: Left Join



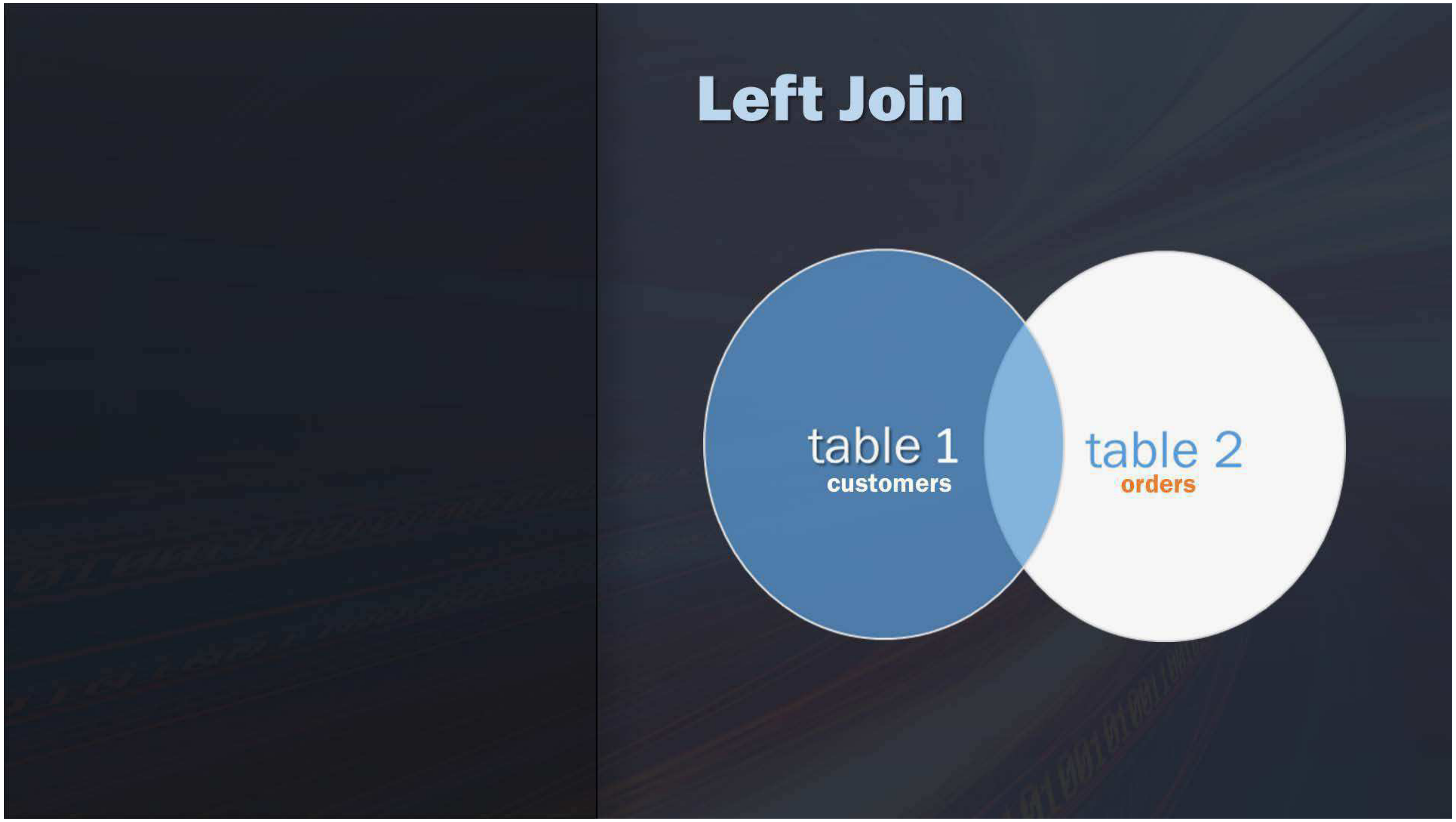
Left Join

Returns all records from the left table (table1), and the matched records from the right table (table2)

The result is NULL from the right side, if there is no match

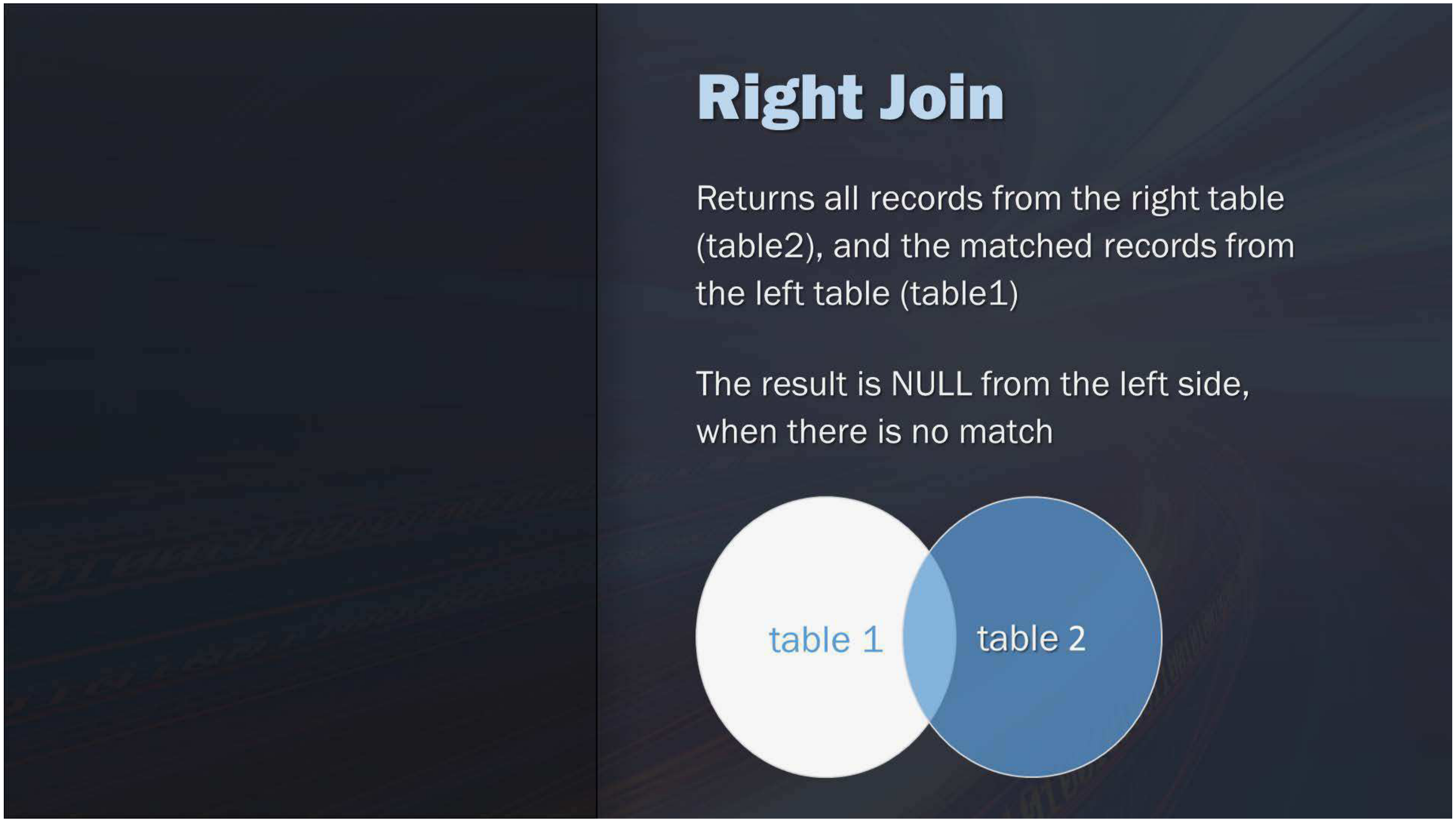
table 1
table 2

Slide 5: Left Join



Left Join

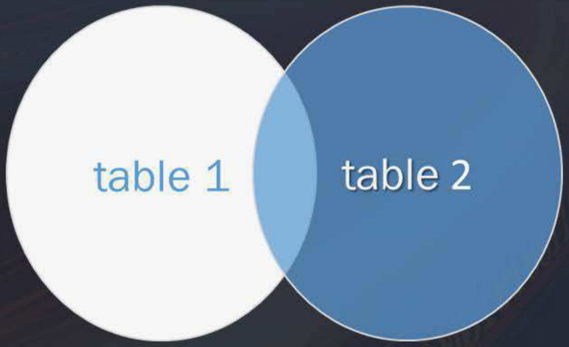
Slide 6: Right Join



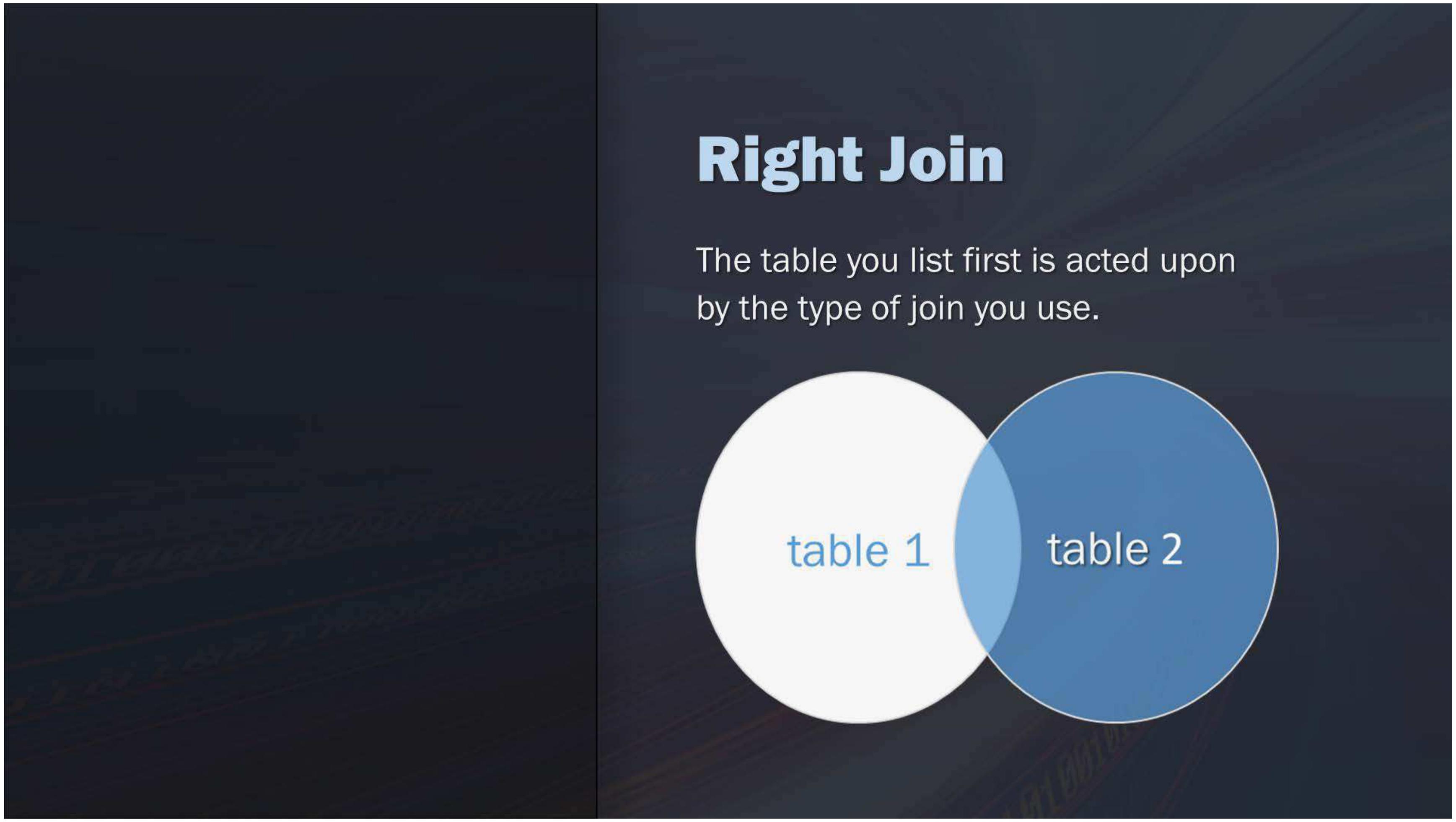
Right Join

Returns all records from the right table (table2), and the matched records from the left table (table1)

The result is NULL from the left side, when there is no match



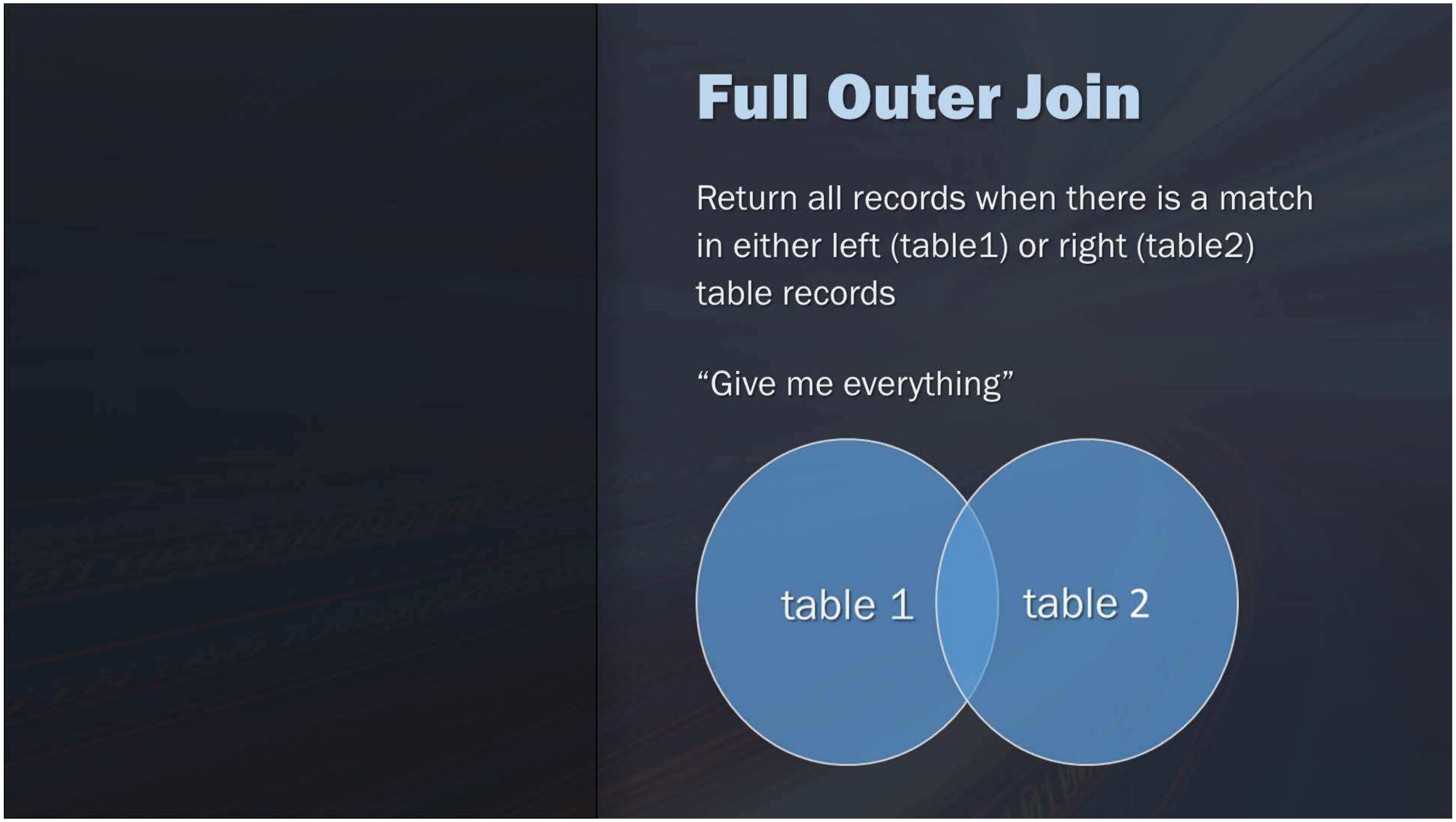
Slide 7: Right Join



Right Join

The table you list first is acted upon by the type of join you use.

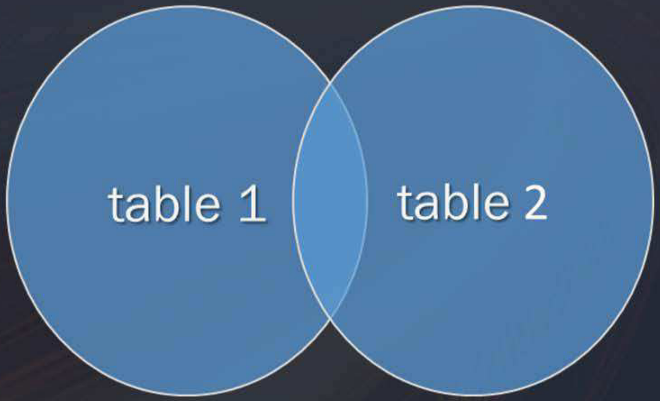
Slide 8: Full Outer Join



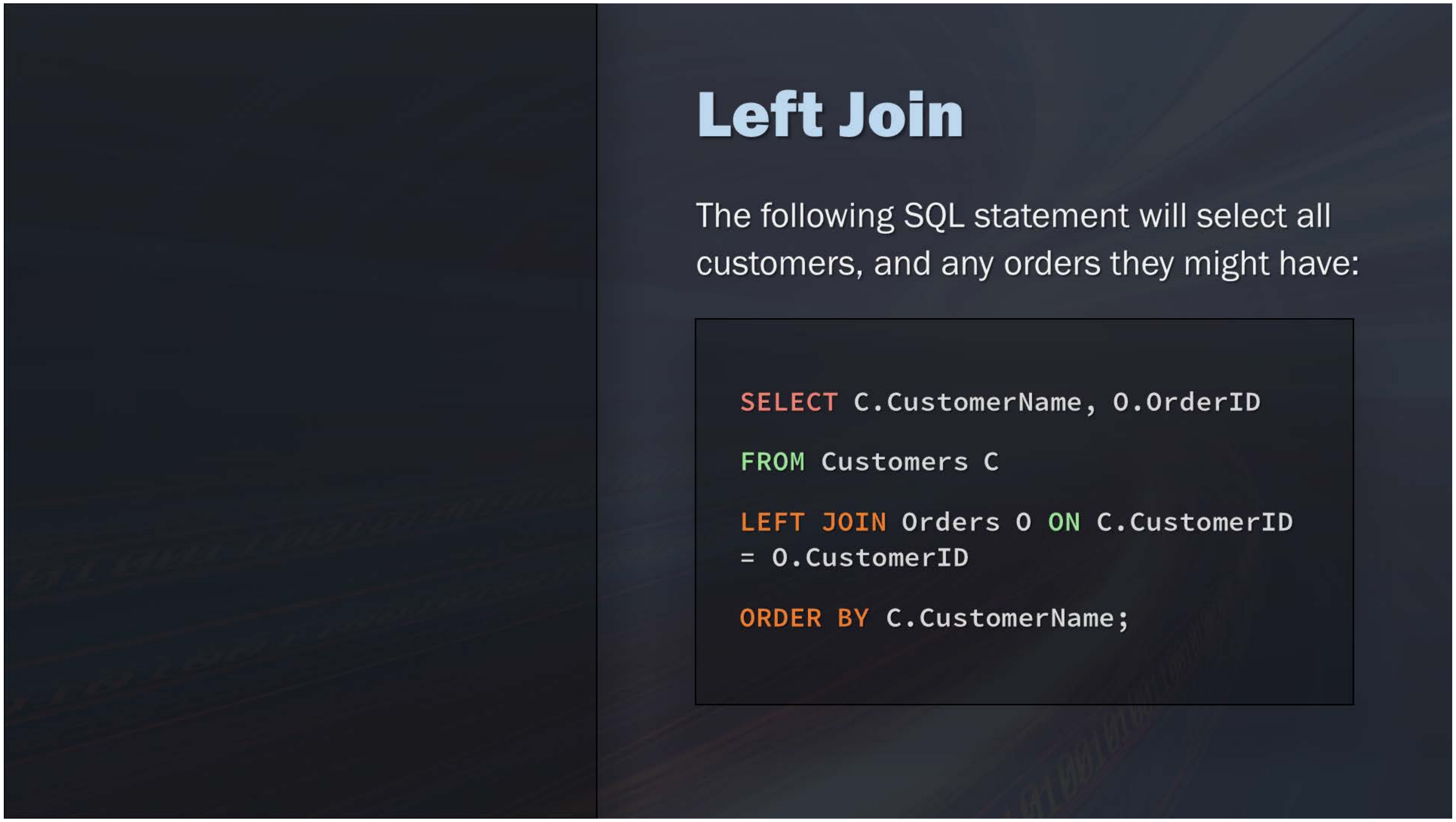
Full Outer Join

Return all records when there is a match in either left (table1) or right (table2) table records

"Give me everything"



Slide 9: Left Join

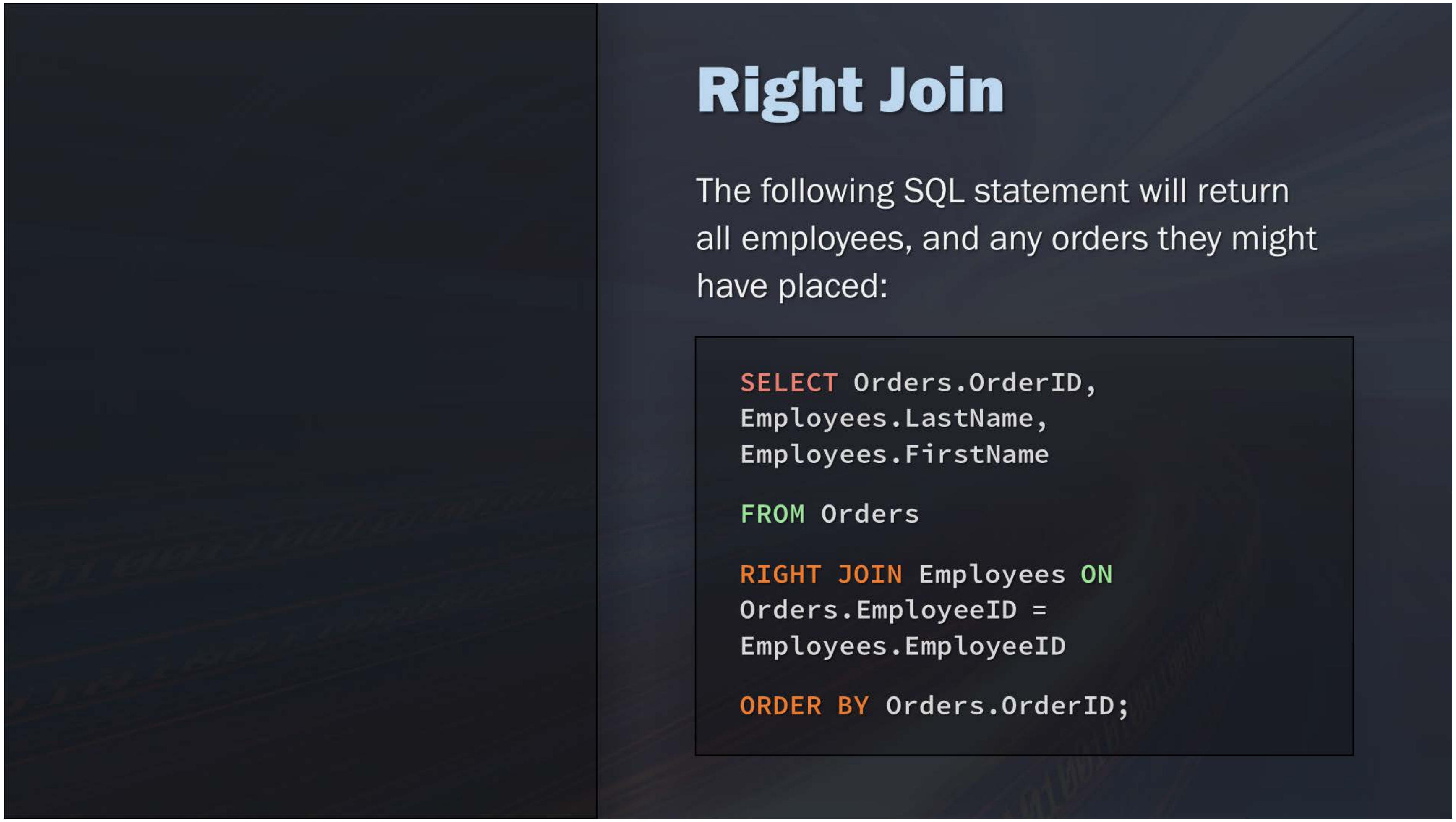


Left Join

The following SQL statement will select all customers, and any orders they might have:

```sql
SELECT C.CustomerName, O.OrderID
FROM Customers C
LEFT JOIN Orders O ON C.CustomerID
= O.CustomerID
ORDER BY C.CustomerName;
```

Slide 10: Right Join



Right Join

The following SQL statement will return all employees, and any orders they might have placed:

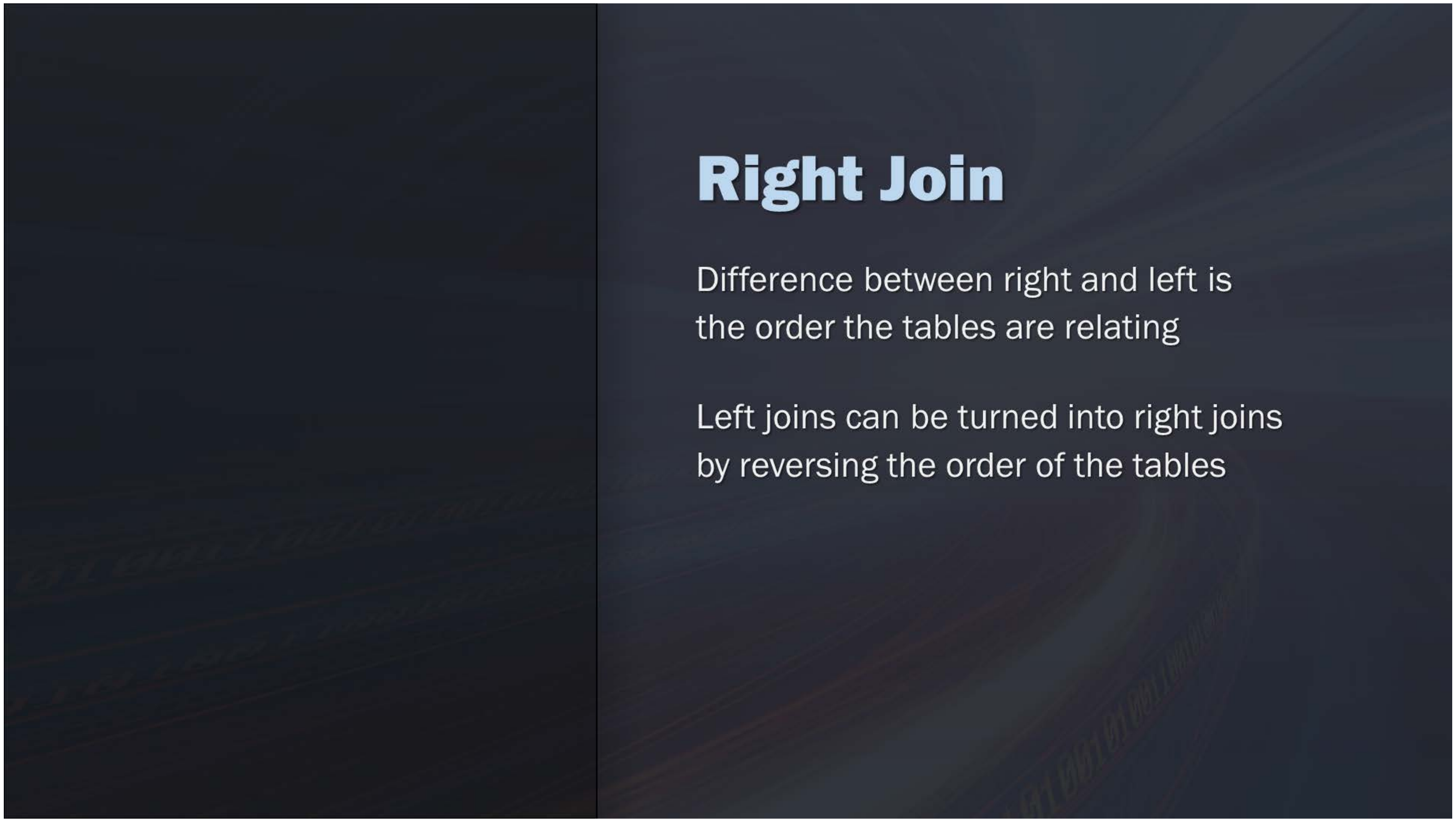
```
SELECT Orders.OrderID,
Employees.LastName,
Employees.FirstName

FROM Orders

RIGHT JOIN Employees ON
Orders.EmployeeID =
Employees.EmployeeID

ORDER BY Orders.OrderID;
```

Slide 11: Right Join

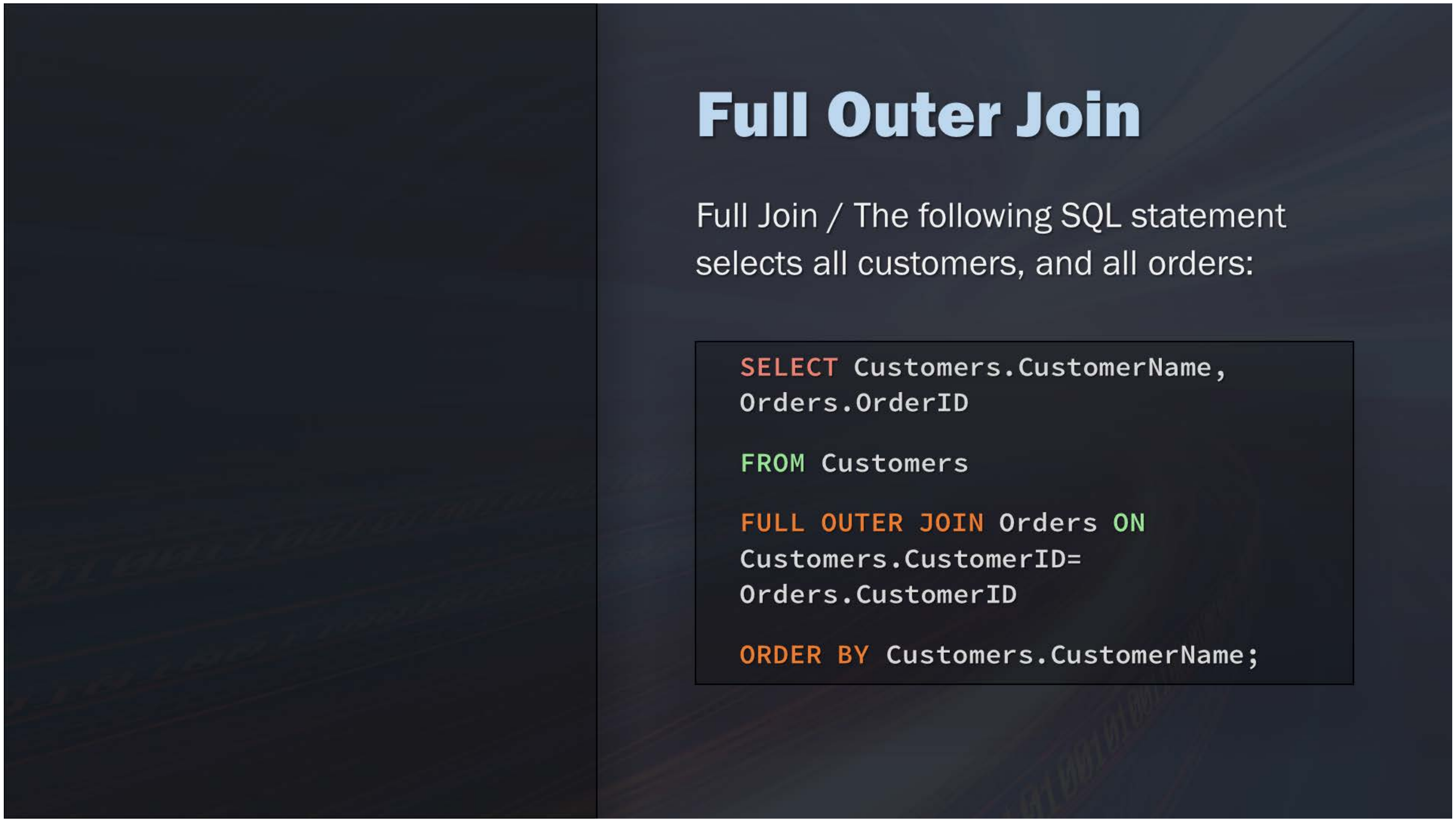


Right Join

Difference between right and left is the order the tables are relating

Left joins can be turned into right joins by reversing the order of the tables

Slide 12: Full Outer Join



Full Outer Join

Full Join / The following SQL statement selects all customers, and all orders:

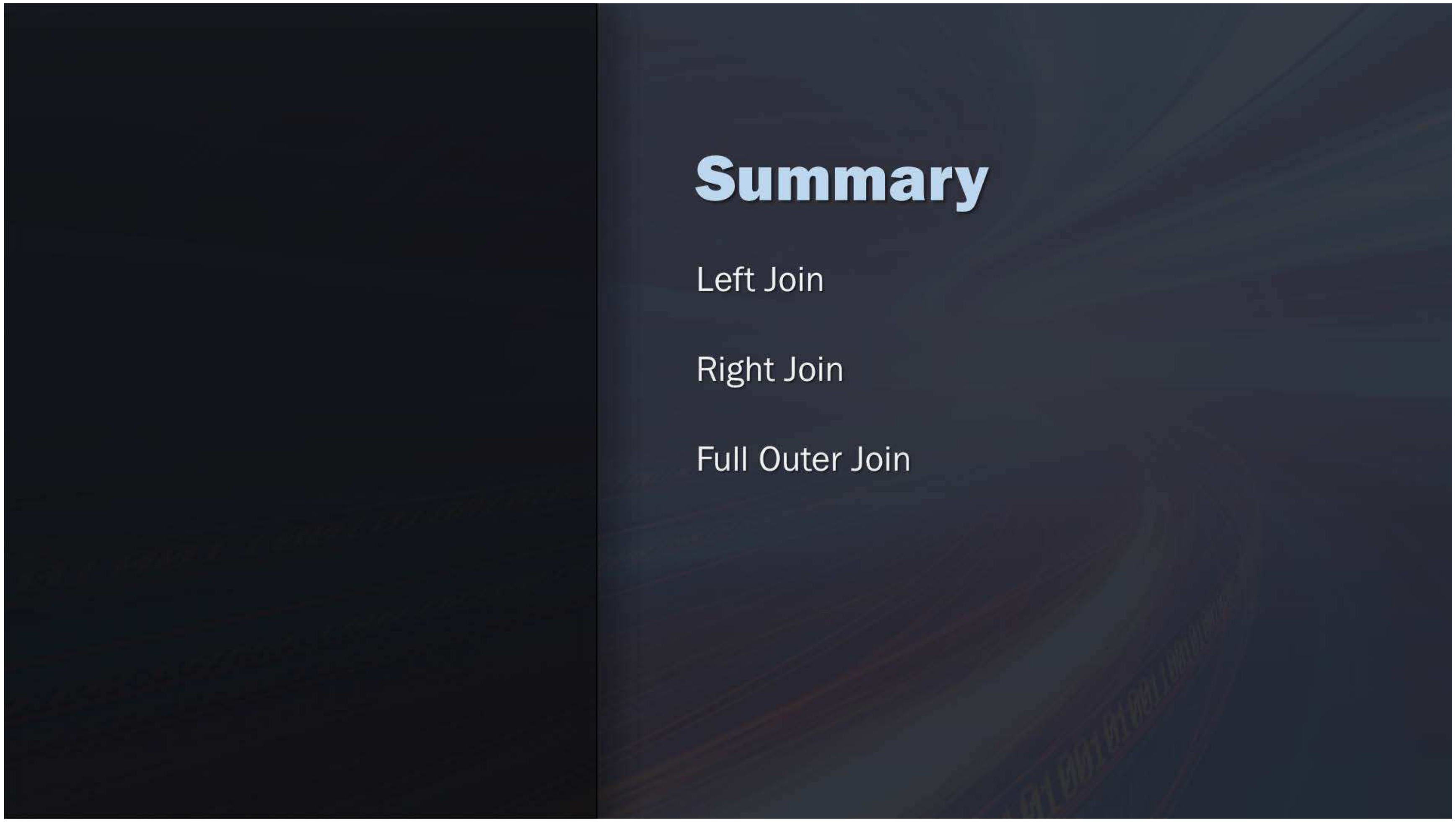
```
SELECT Customers.CustomerName,
Orders.OrderID

FROM Customers

FULL OUTER JOIN Orders ON
Customers.CustomerID=
Orders.CustomerID

ORDER BY Customers.CustomerName;
```

Slide 13: Summary



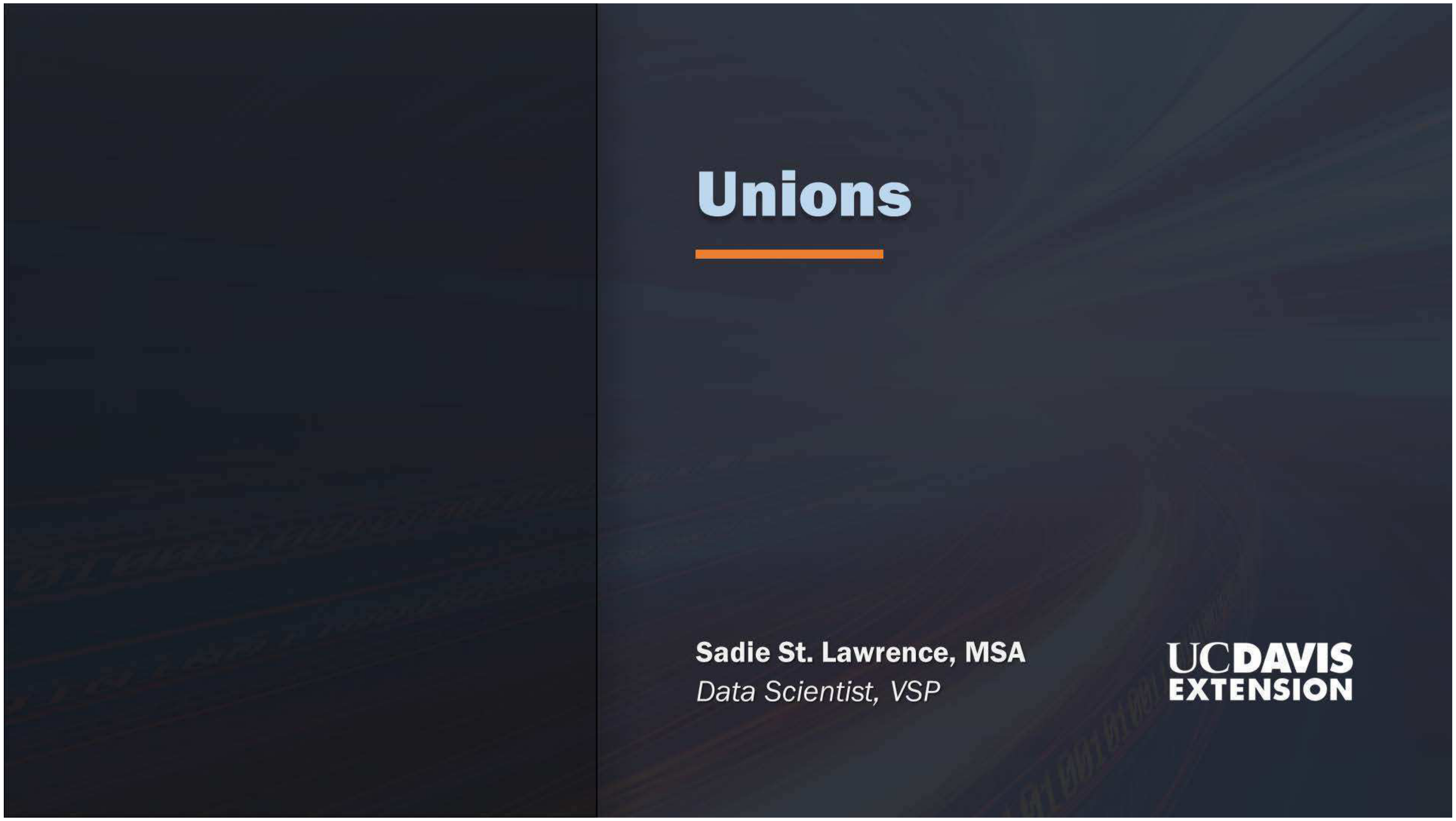
Summary

Left Join

Right Join

Full Outer Join

Slide 1: Unions

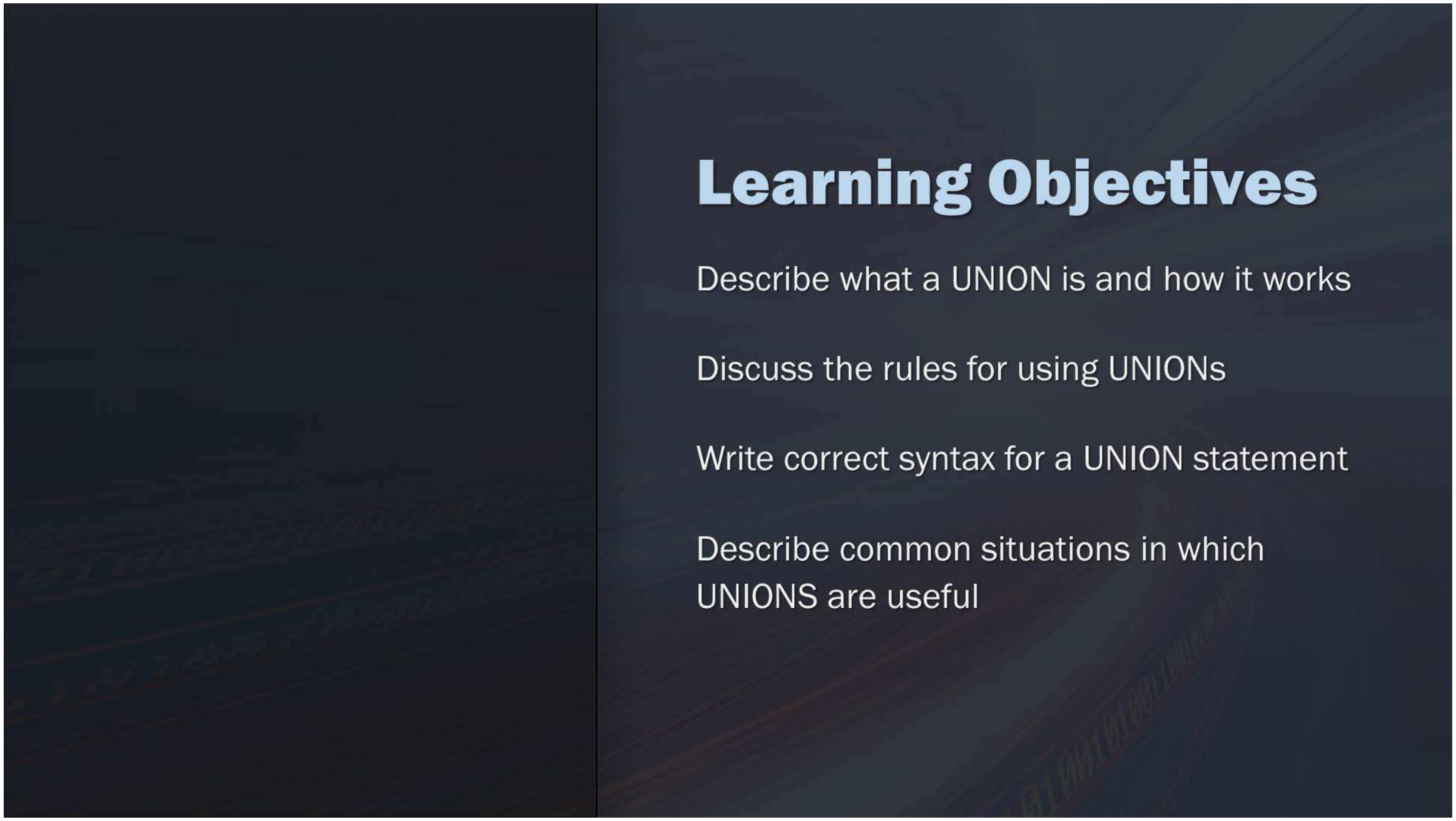


Unions

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Learning Objectives



Learning Objectives

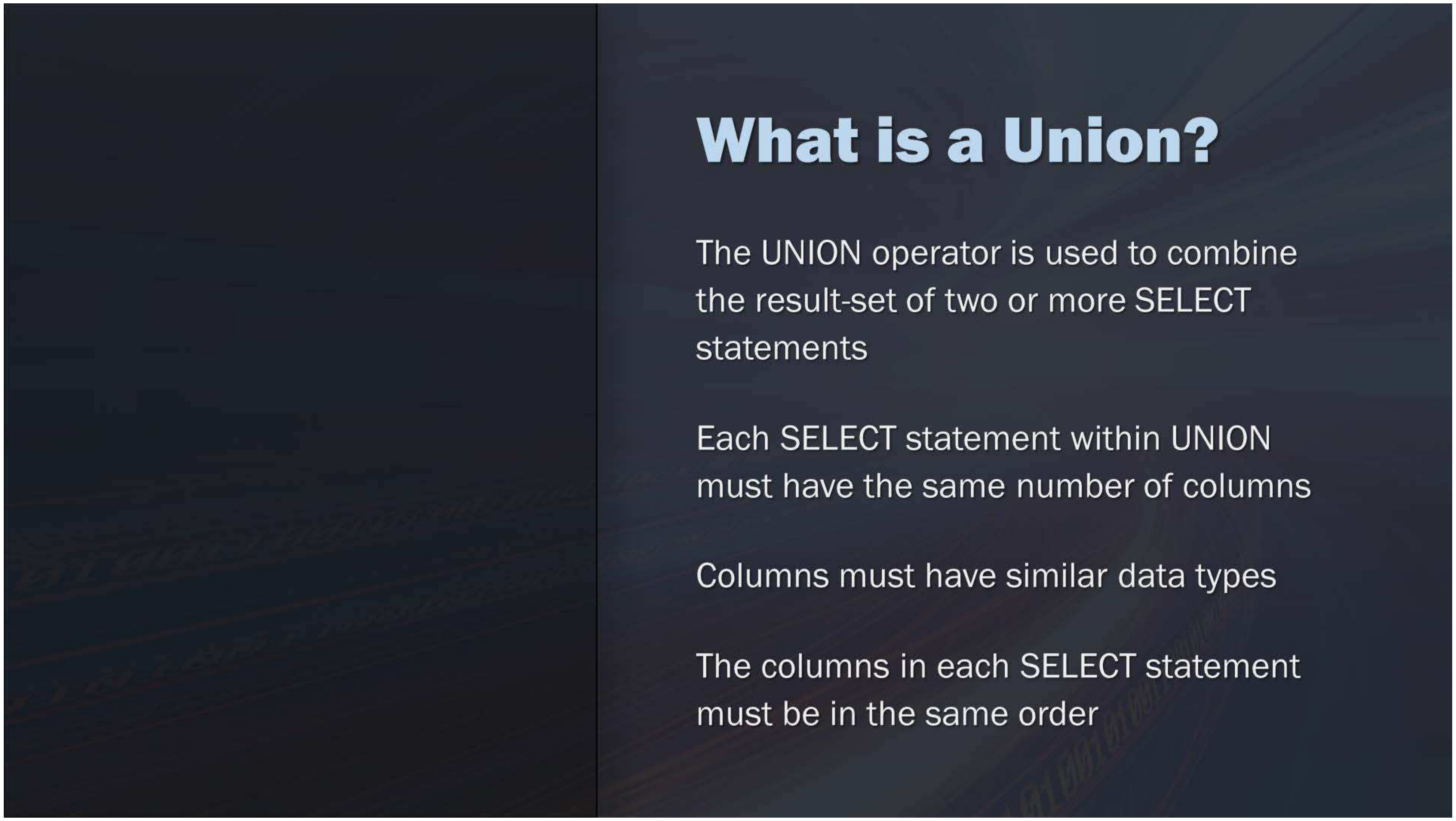
Describe what a UNION is and how it works

Discuss the rules for using UNIONs

Write correct syntax for a UNION statement

Describe common situations in which UNIONS are useful

Slide 3: What is a Union



What is a Union?

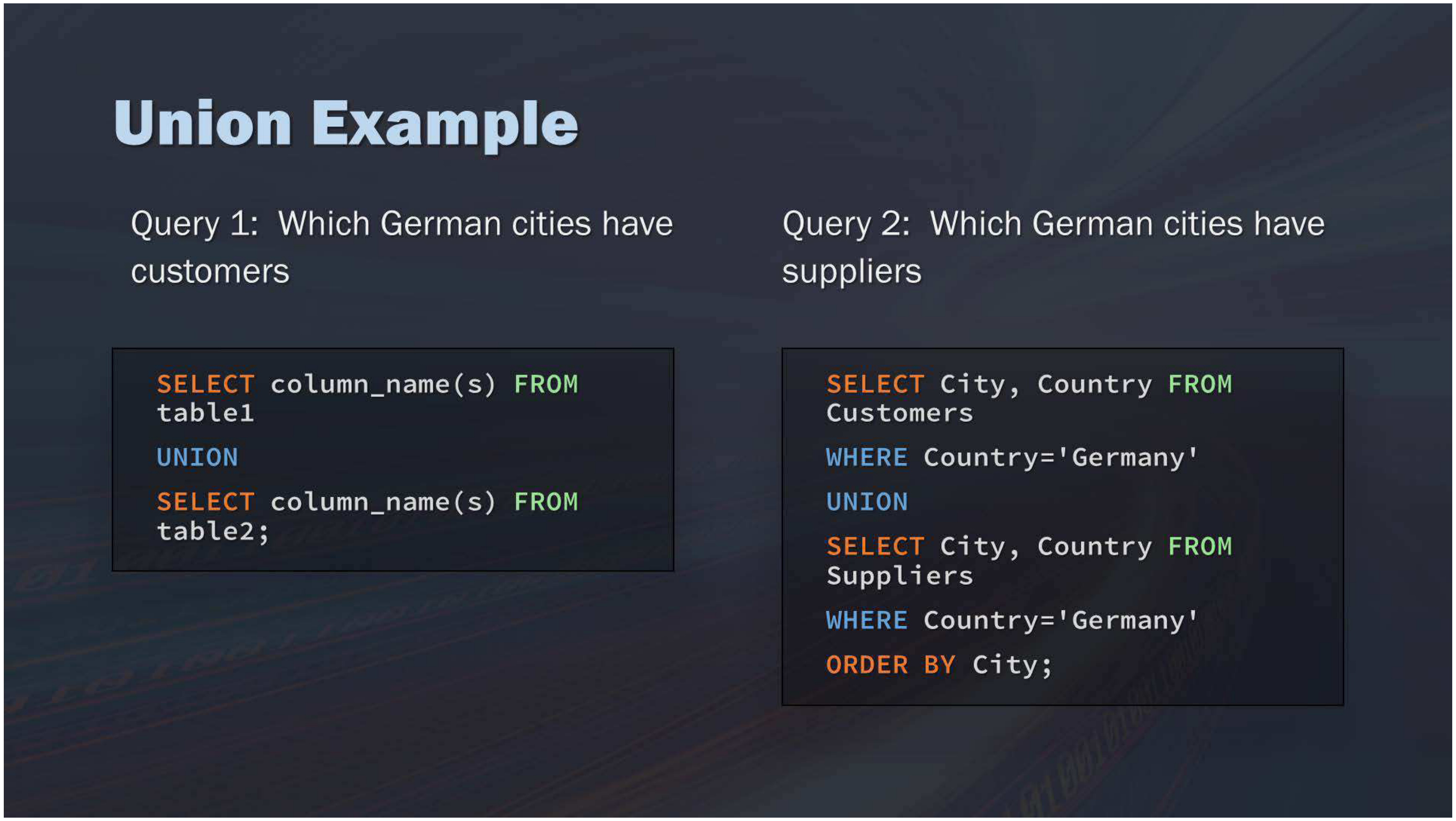
The UNION operator is used to combine the result-set of two or more SELECT statements

Each SELECT statement within UNION must have the same number of columns

Columns must have similar data types

The columns in each SELECT statement must be in the same order

Slide 4: Union Example



Union Example

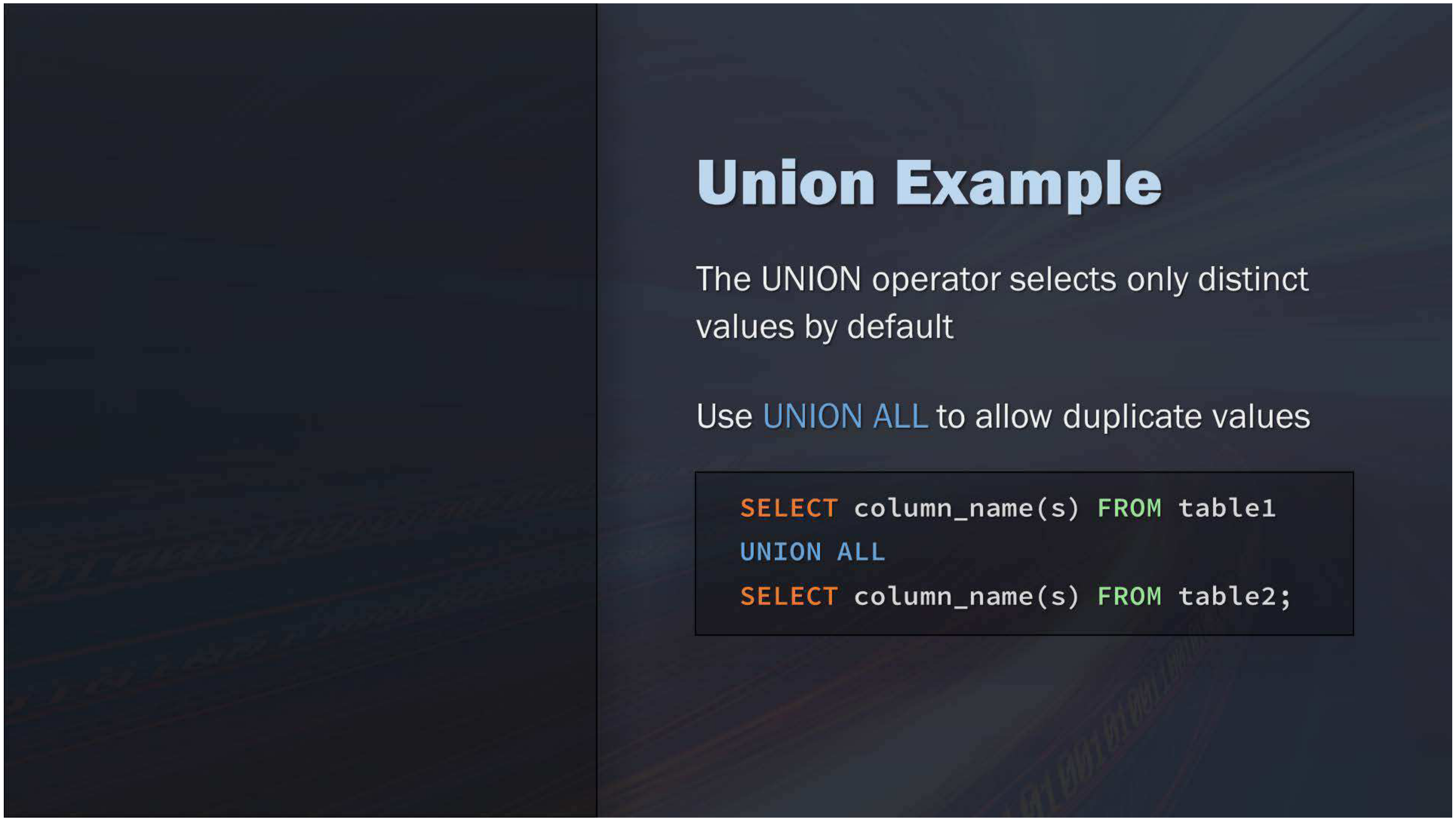
Query 1: Which German cities have customers

```
SELECT column_name(s) FROM
table1
UNION
SELECT column_name(s) FROM
table2;
```

Query 2: Which German cities have suppliers

```
SELECT City, Country FROM
Customers
WHERE Country='Germany'
UNION
SELECT City, Country FROM
Suppliers
WHERE Country='Germany'
ORDER BY City;
```

Slide 5: Union Example



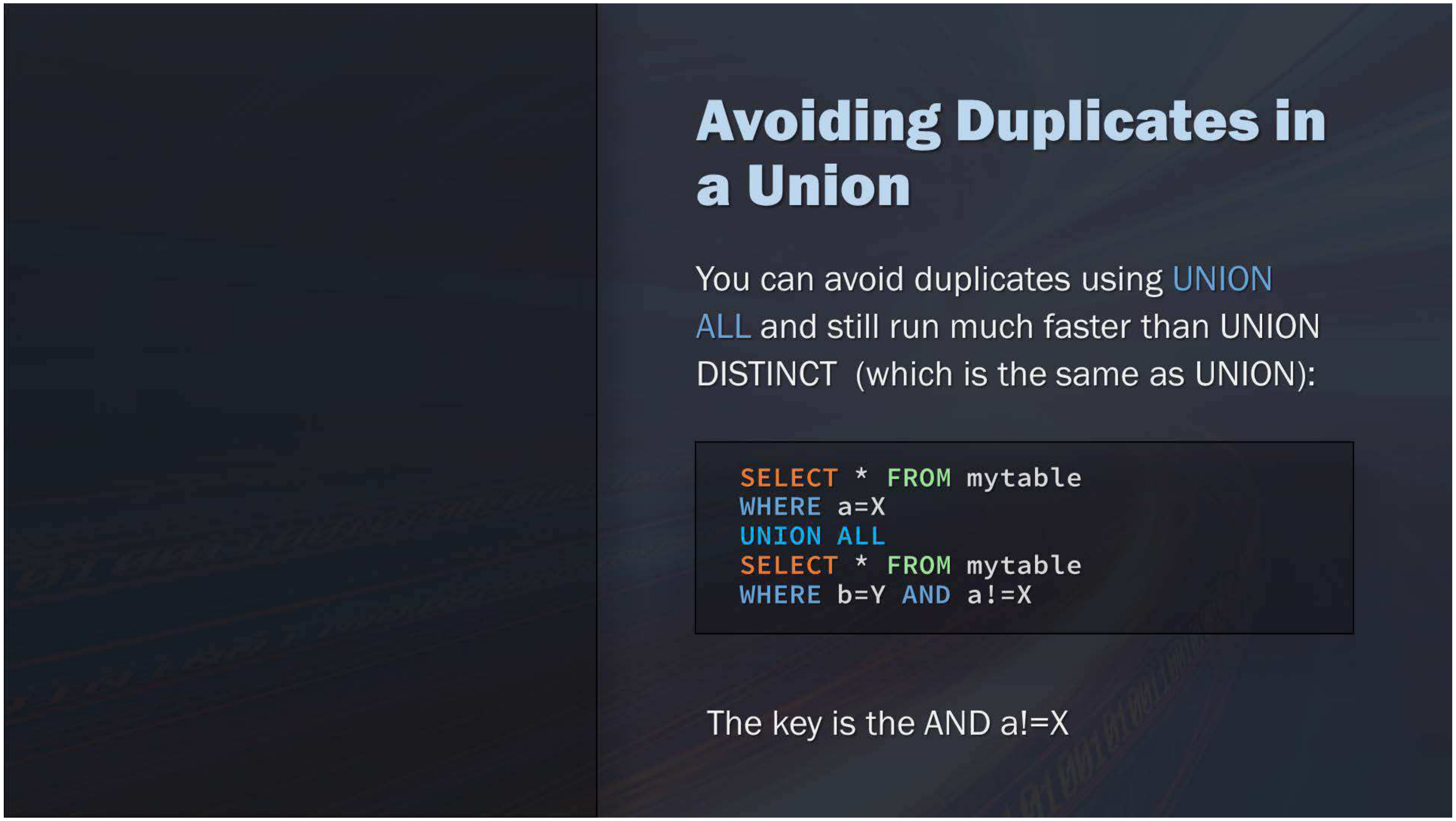
Union Example

The UNION operator selects only distinct values by default

Use UNION ALL to allow duplicate values

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

Slide 6: Avoiding Duplicates in a Union



Avoiding Duplicates in a Union

You can avoid duplicates using UNION ALL and still run much faster than UNION DISTINCT (which is the same as UNION):

```
SELECT * FROM mytable
WHERE a=X
UNION ALL
SELECT * FROM mytable
WHERE b=Y AND a!=X
```

The key is the AND a!=X

Slide 1: Summary

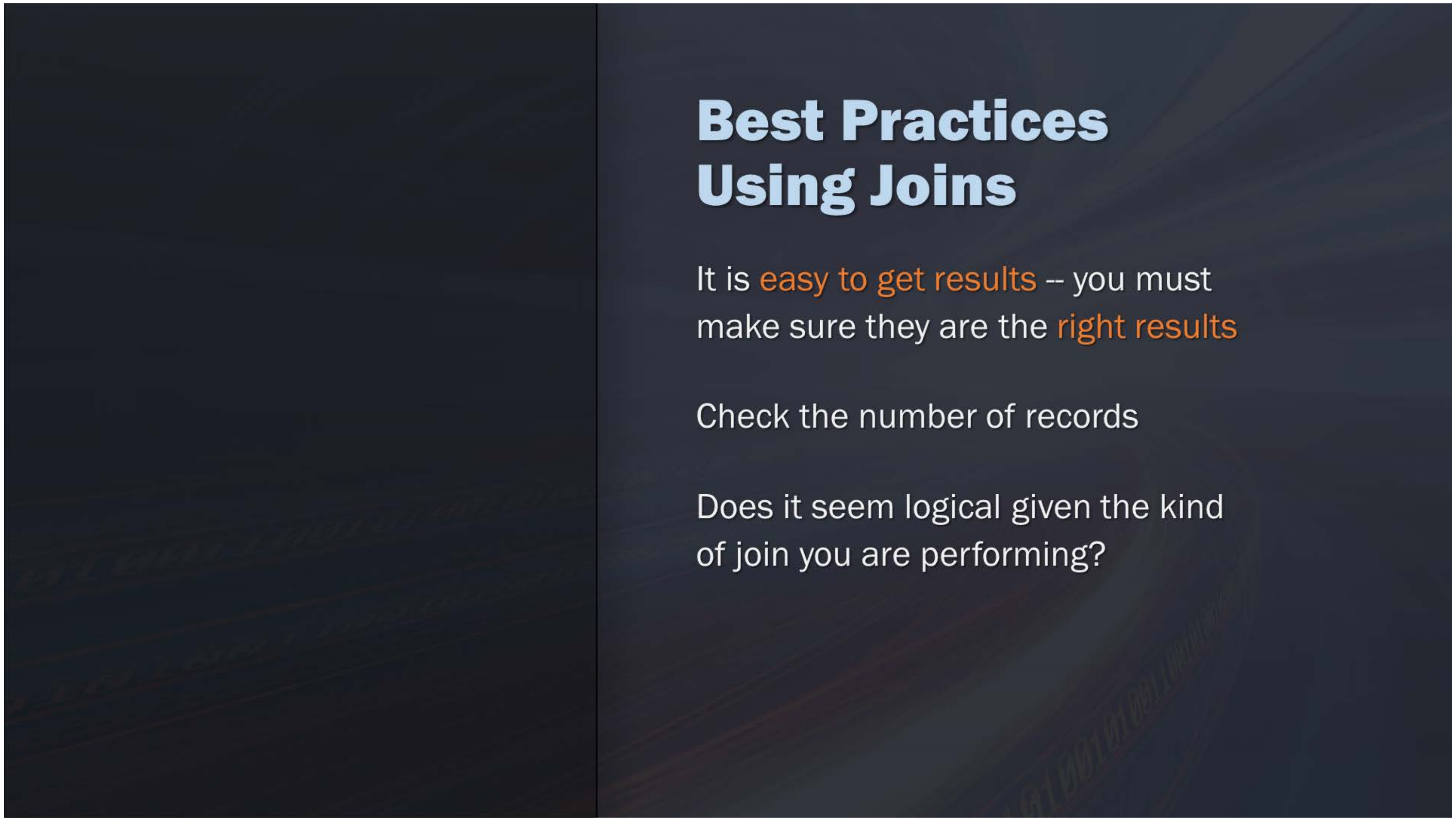
Summary

Sadie St. Lawrence, MSA

Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Best Practices Using Joins



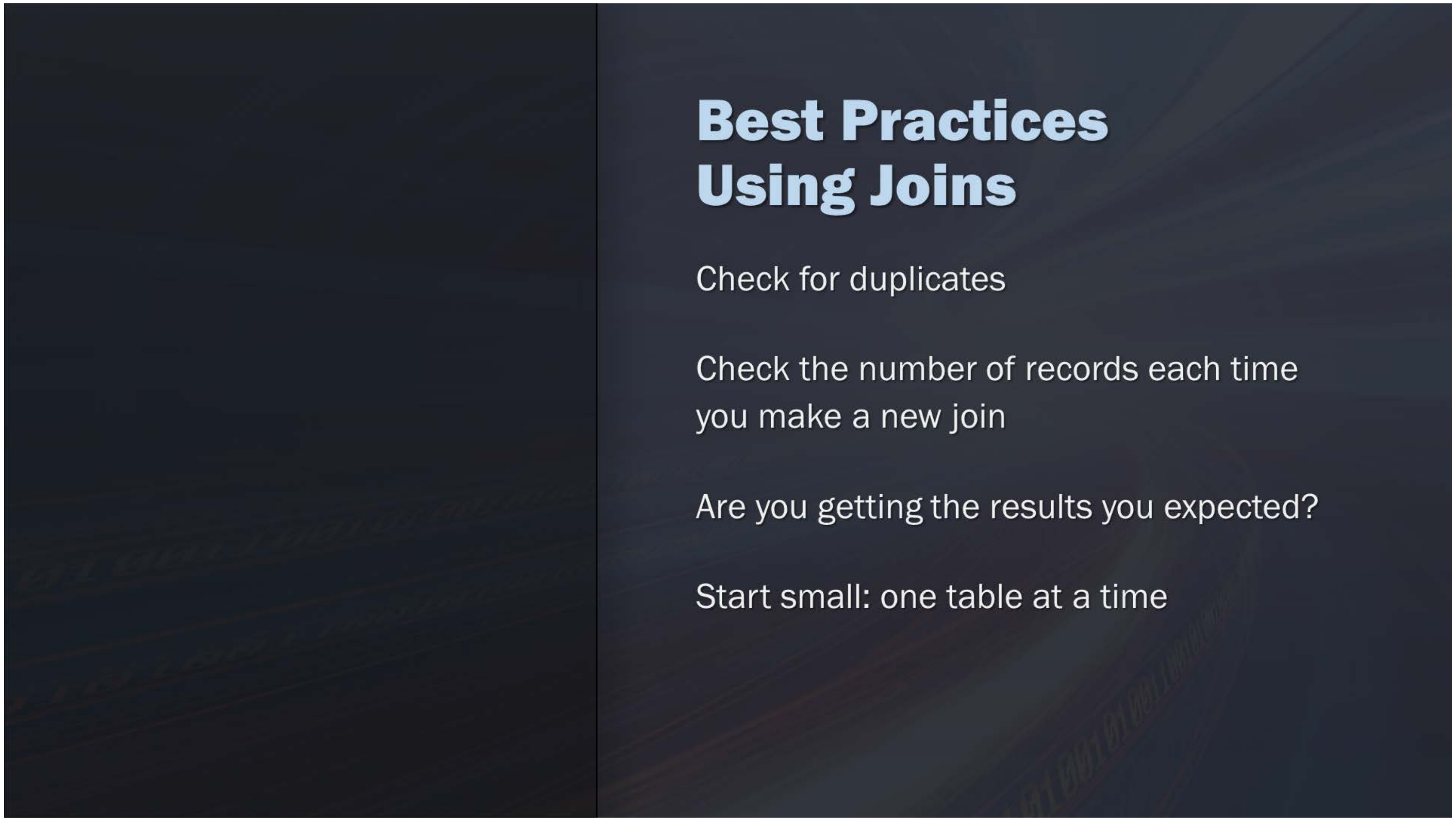
Best Practices Using Joins

It is easy to get results -- you must make sure they are the right results

Check the number of records

Does it seem logical given the kind of join you are performing?

Slide 3: Best Practices Using Joins



Best Practices Using Joins

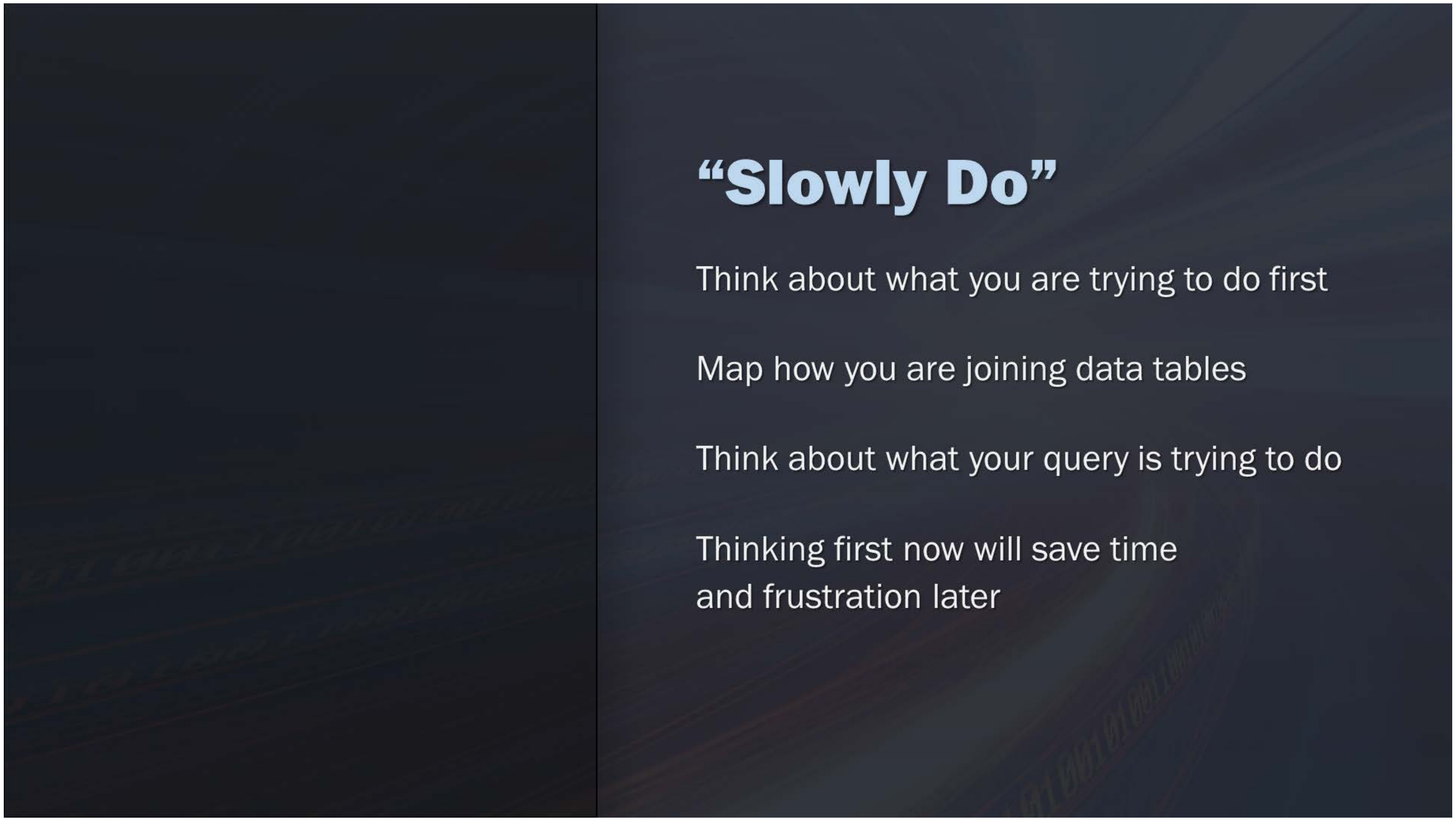
Check for duplicates

Check the number of records each time you make a new join

Are you getting the results you expected?

Start small: one table at a time

Slide 4: “Slowly Do”



“Slowly Do”

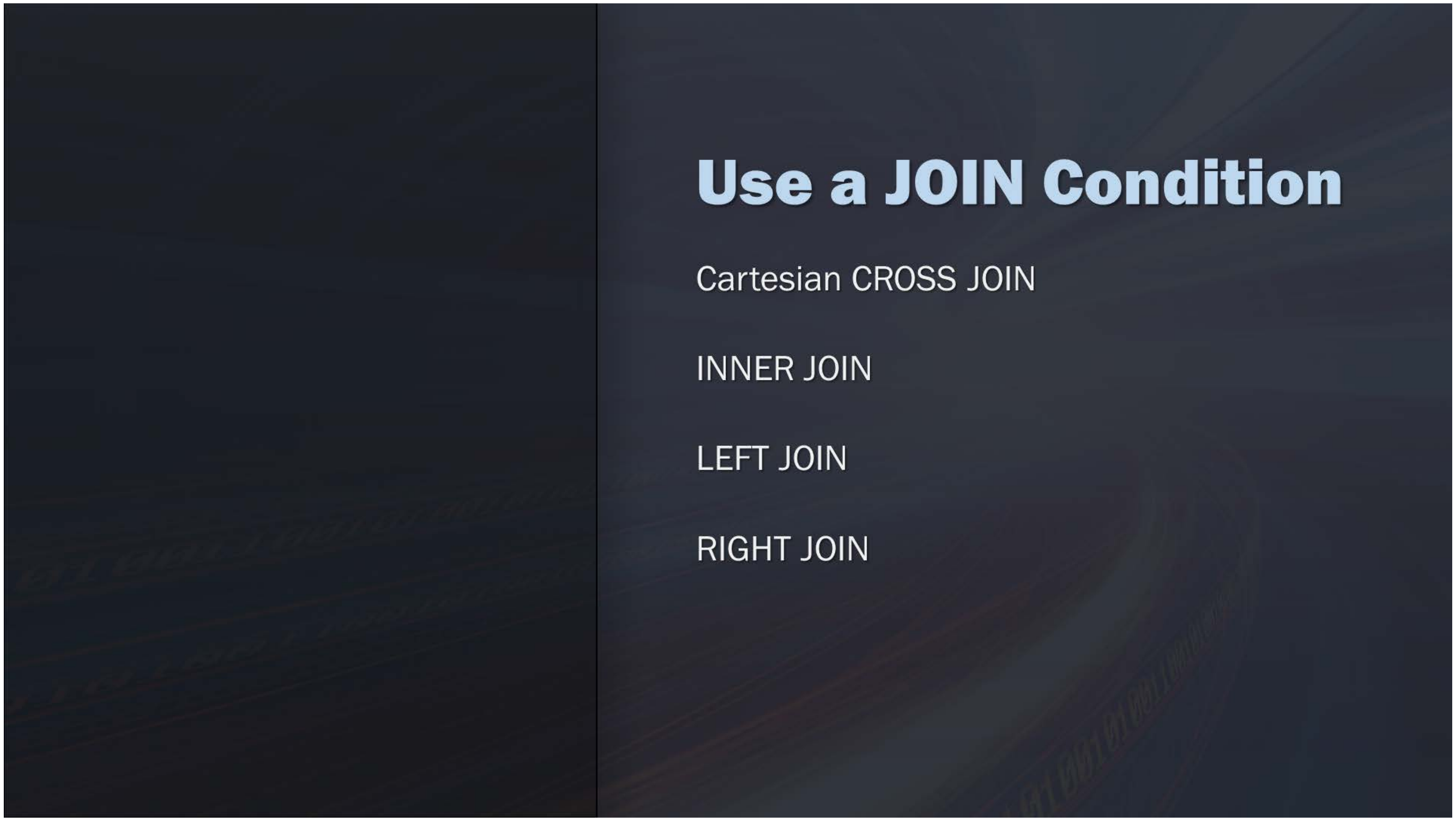
Think about what you are trying to do first

Map how you are joining data tables

Think about what your query is trying to do

Thinking first now will save time
and frustration later

Slide 5: Use a JOIN Condition



Use a JOIN Condition

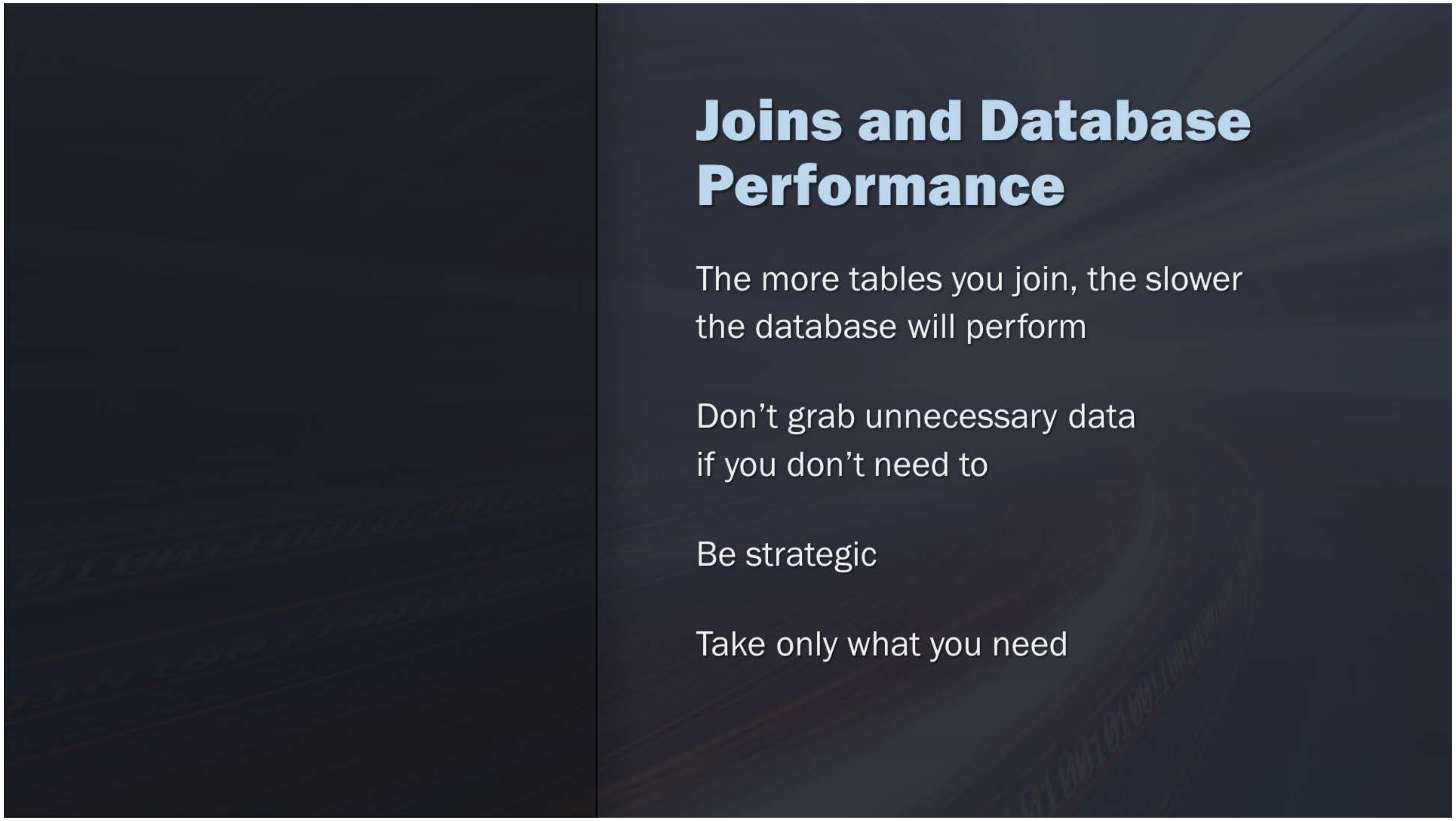
Cartesian CROSS JOIN

INNER JOIN

LEFT JOIN

RIGHT JOIN

Slide 6: Joins and Database Performance



Joins and Database Performance

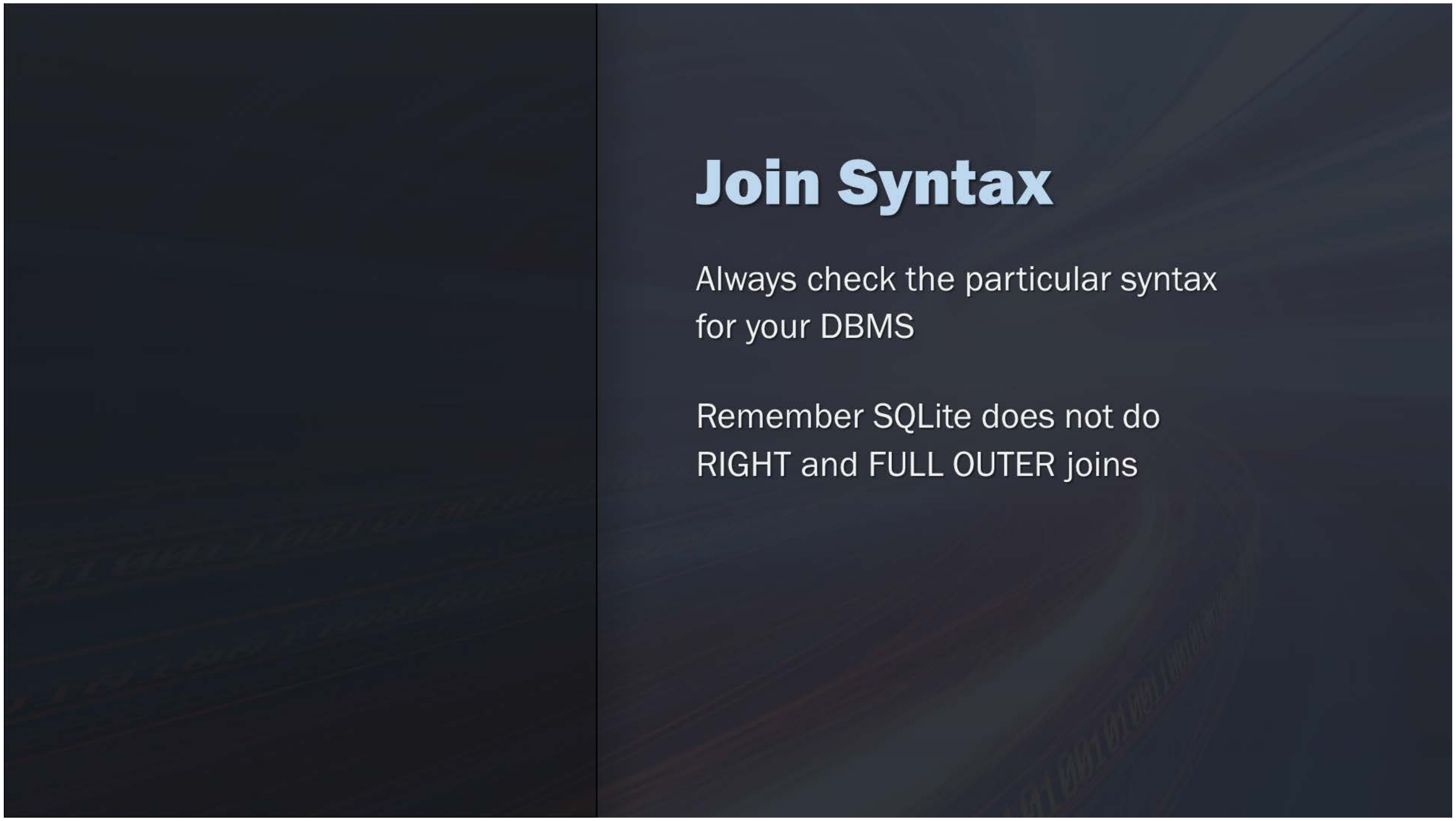
The more tables you join, the slower the database will perform

Don't grab unnecessary data if you don't need to

Be strategic

Take only what you need

Slide 7: Join Syntax

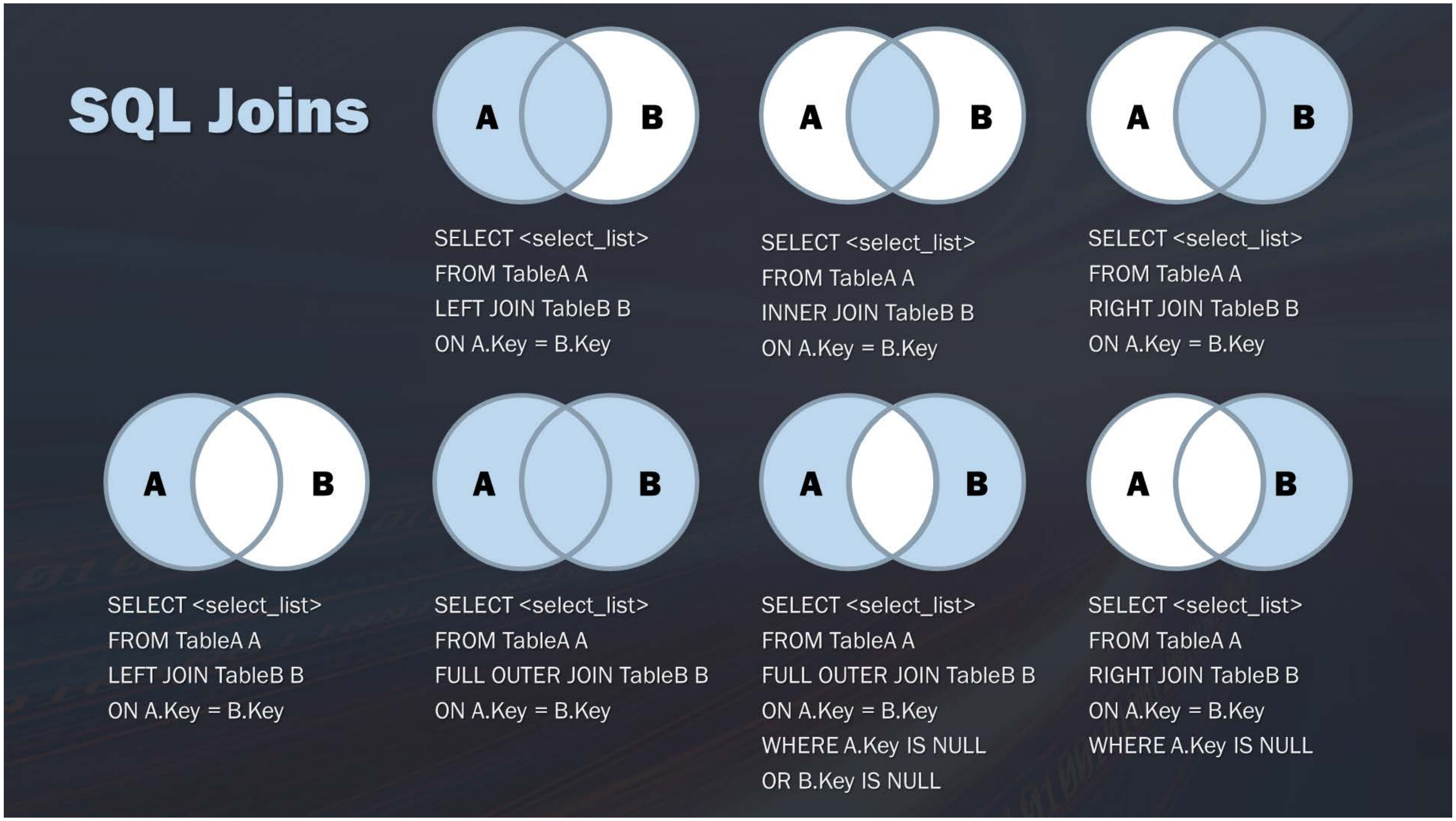


Join Syntax

Always check the particular syntax for your DBMS

Remember SQLite does not do RIGHT and FULL OUTER joins

Slide 8: SQL Joins



SQL Joins

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

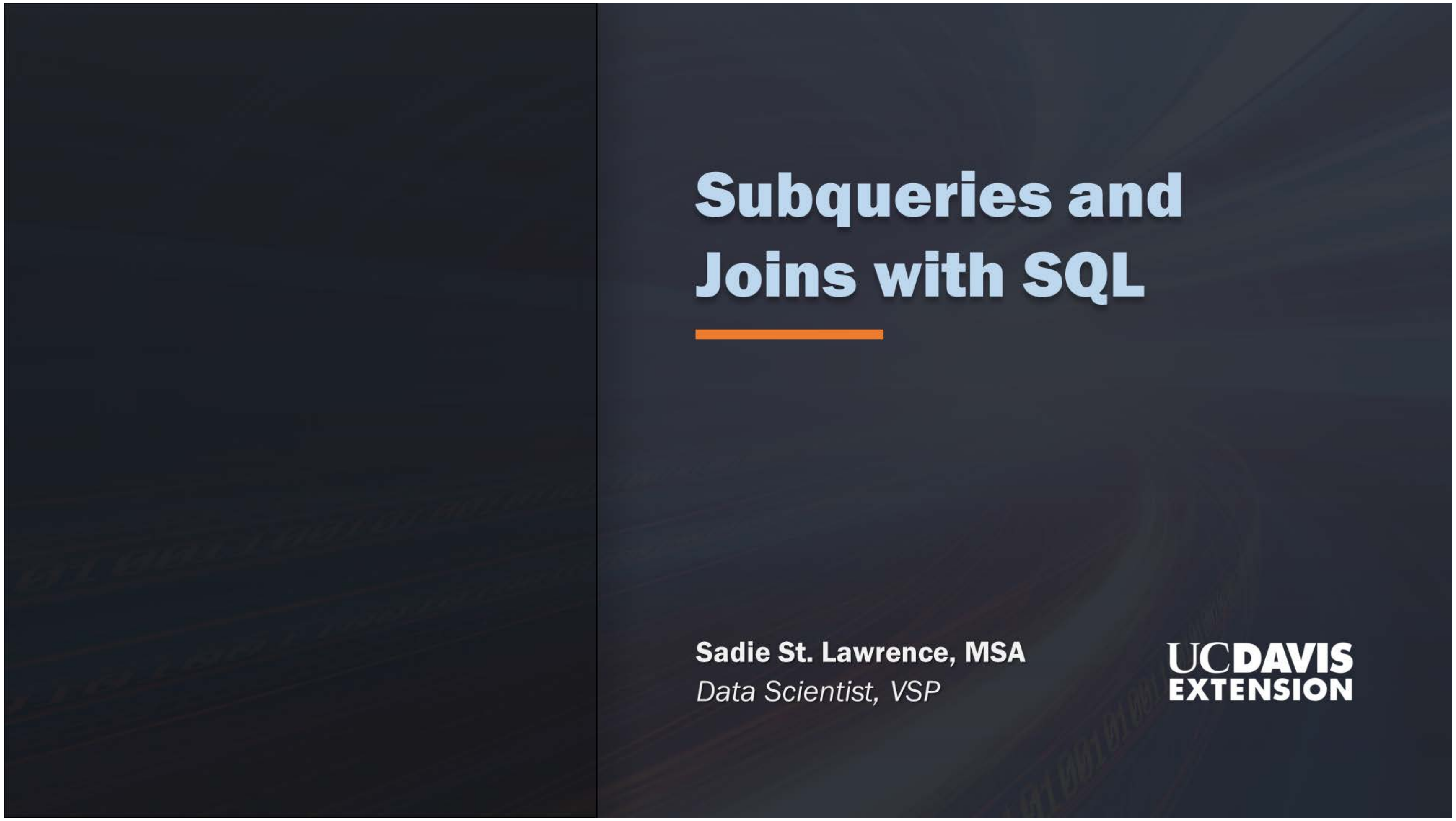
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

Slide 1: Subqueries and Joins with SQL

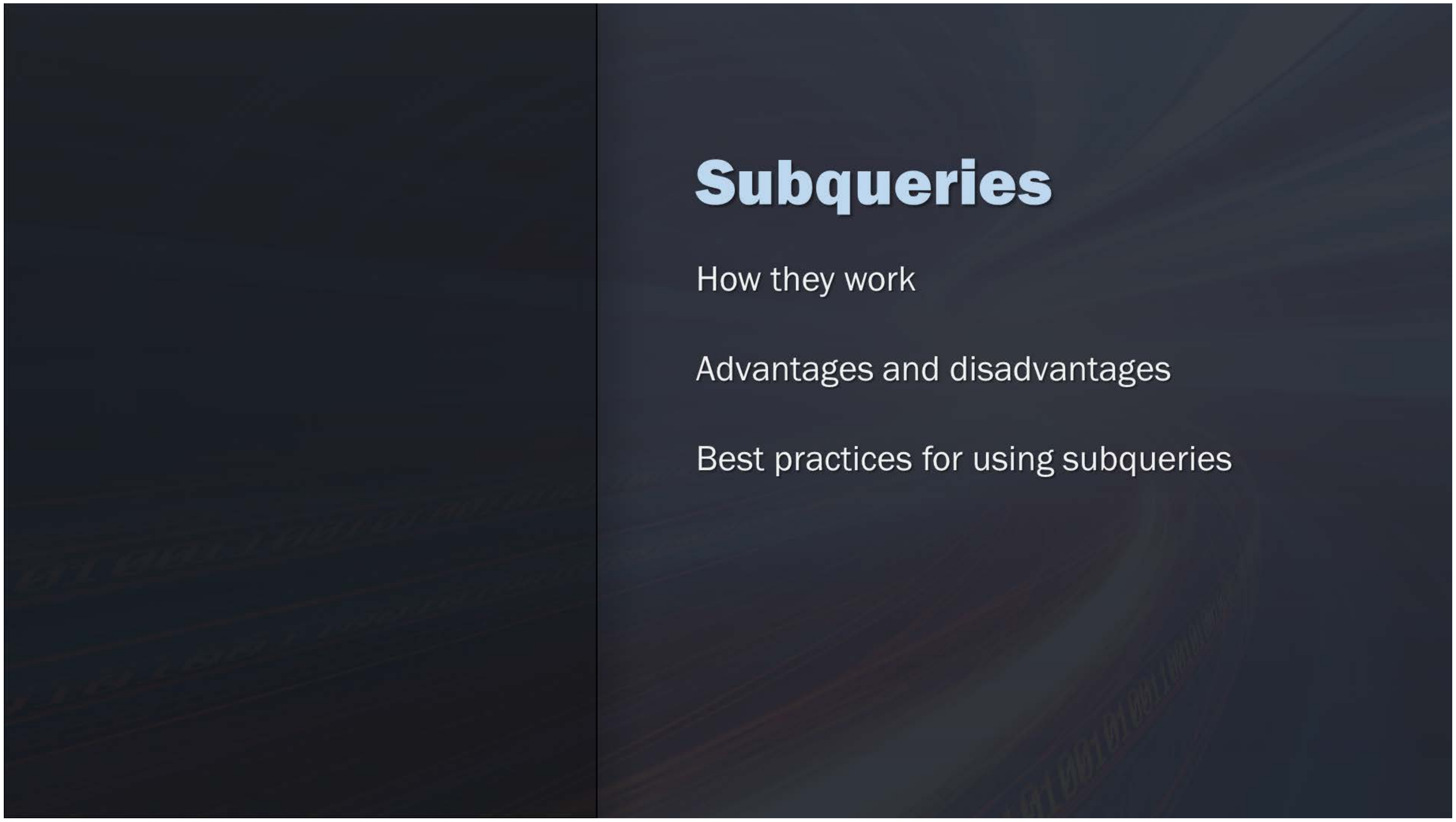


Subqueries and Joins with SQL

Sadie St. Lawrence, MSA
Data Scientist, VSP

UCDAVIS EXTENSION

Slide 2: Subqueries



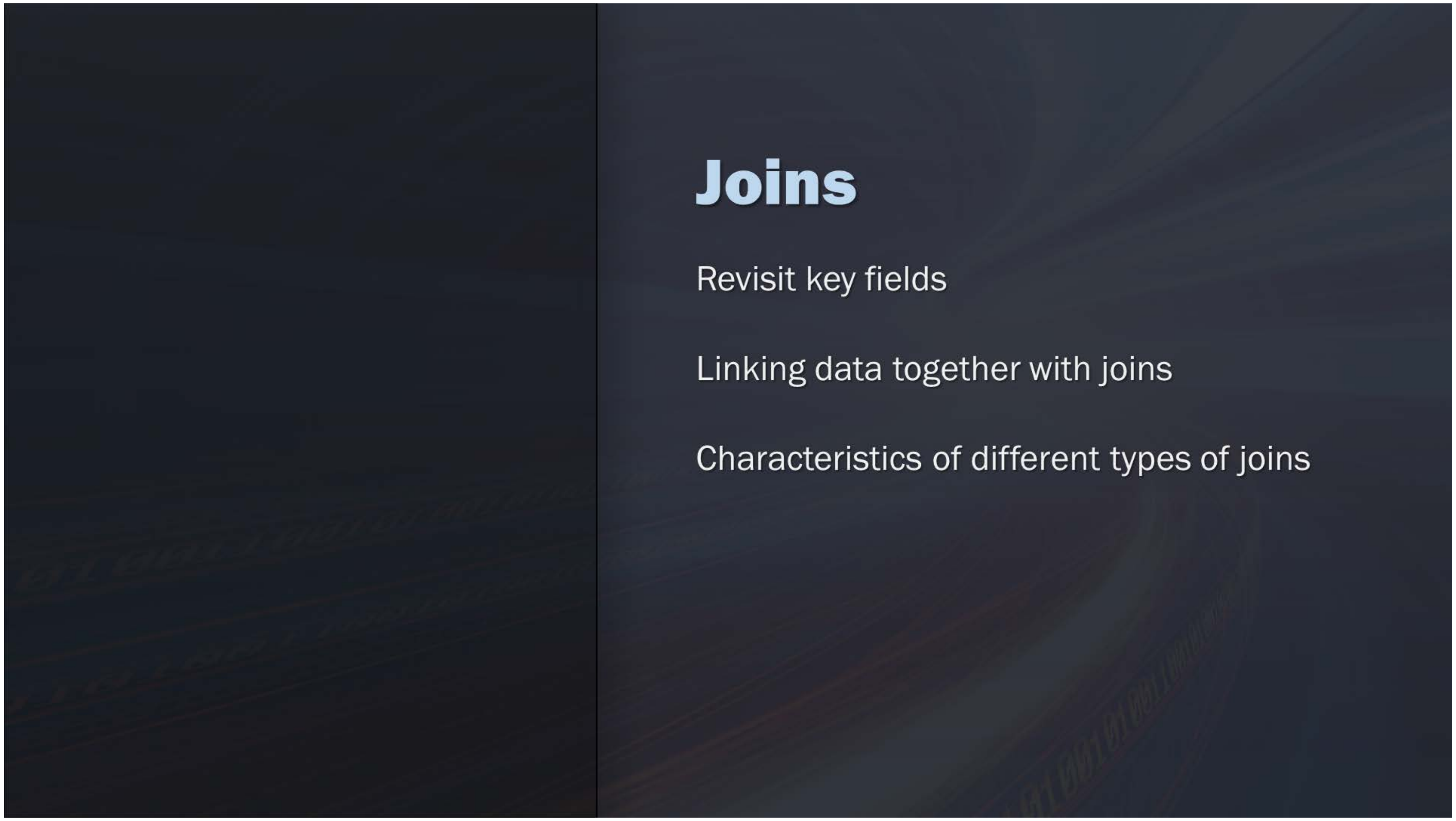
Subqueries

How they work

Advantages and disadvantages

Best practices for using subqueries

Slide 3: Joins



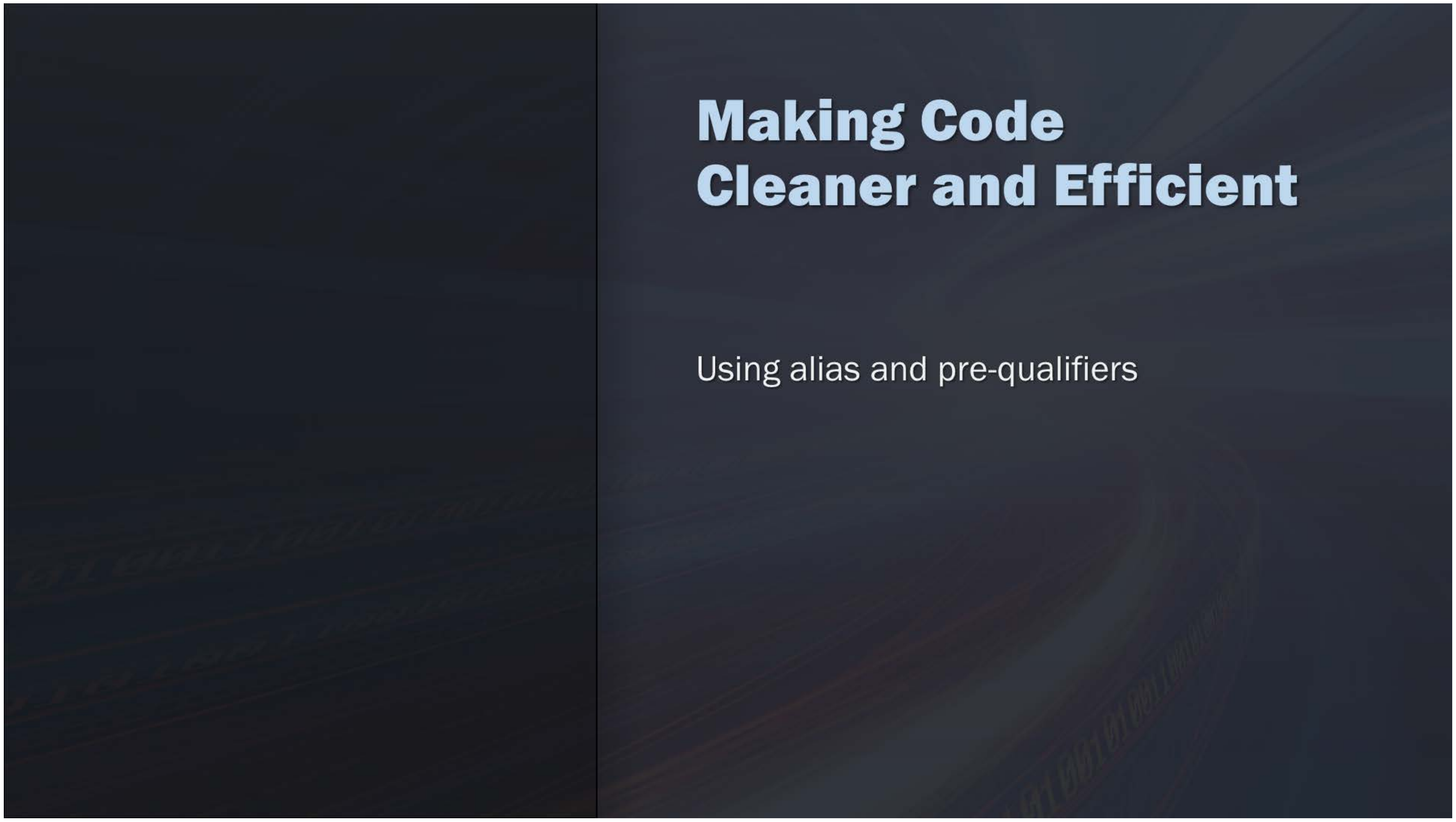
Joins

Revisit key fields

Linking data together with joins

Characteristics of different types of joins

Slide 4: Making Code Cleaner and Efficient



Making Code Cleaner and Efficient

Using alias and pre-qualifiers