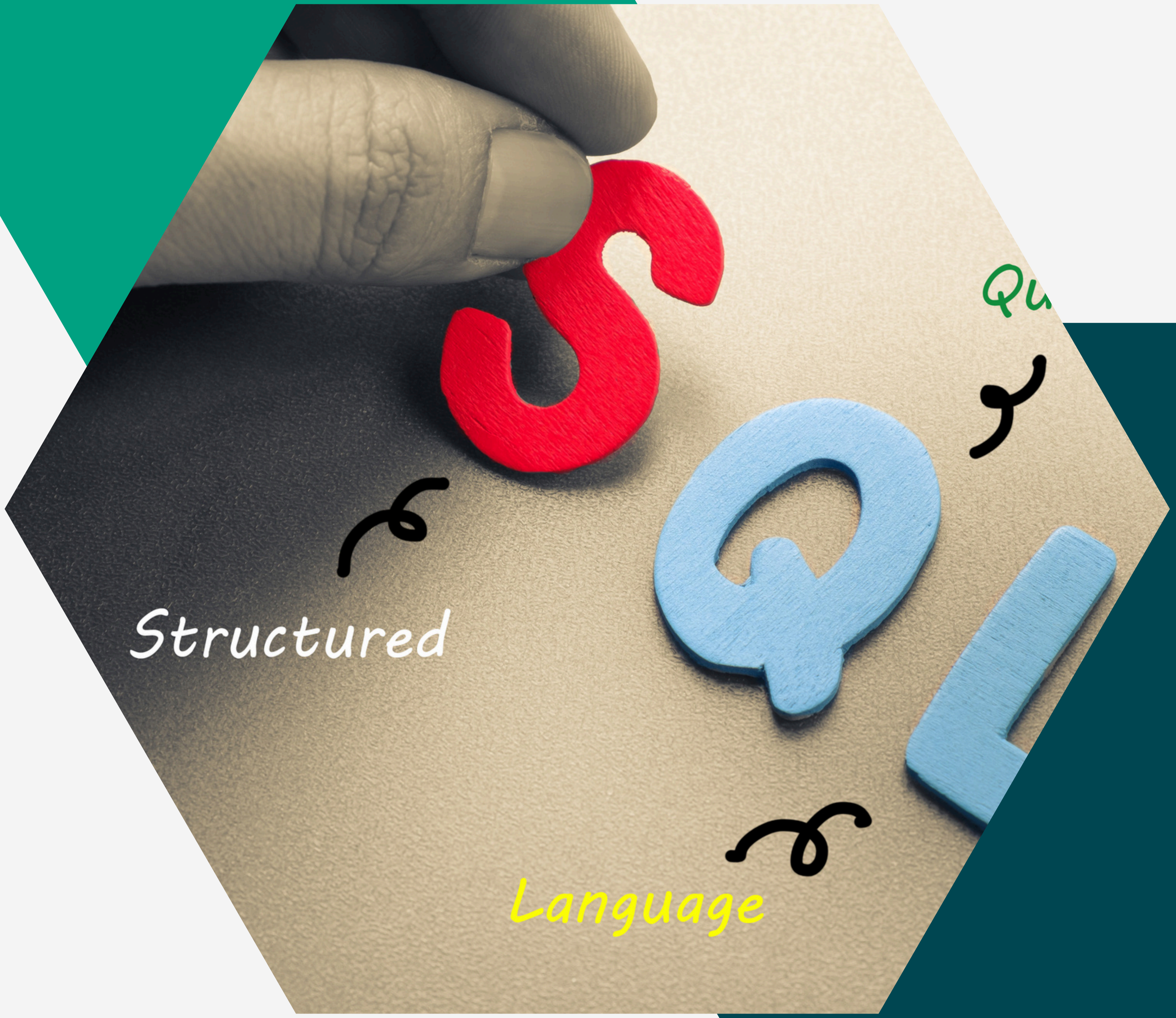
Introduction to SQL

Agenda

- What is SQL
- Basics of SQL - CRUD
- Data Types
- Select Statements & Where Clause
- String Functions
- Aggregate Functions
- Group By & Having Clause
- Date Time Funtions
- SQL Joins
- SQL Case Statements
- Download MySQL Workbench

What is SQL?

Structured Query Language (SQL) is a programming language that helps you interact with databases



Basics of SQL - CRUD

CREATE

means adding or inserting rows into a table

READ

means adding or inserting rows into a table

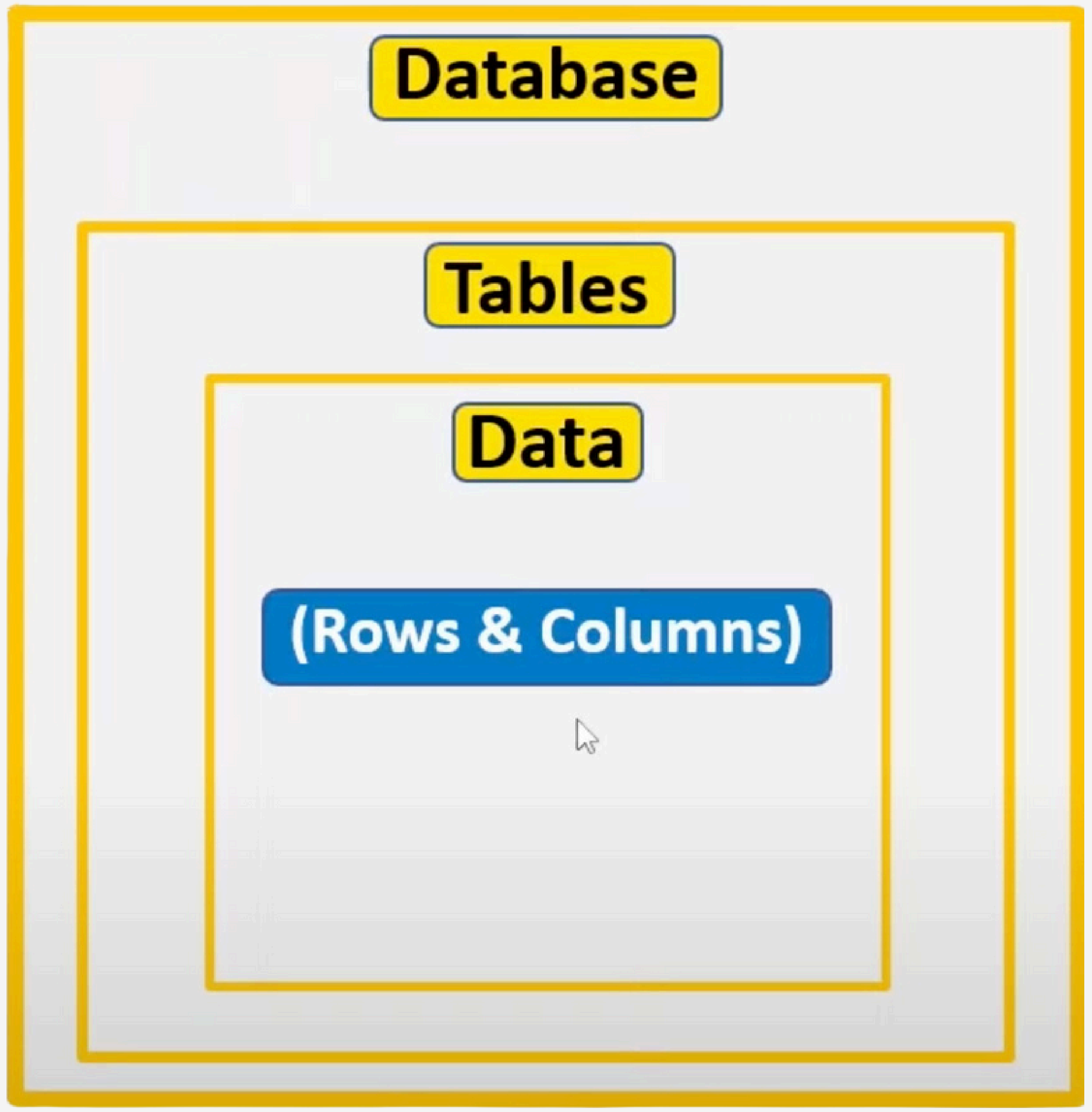
UPDATE

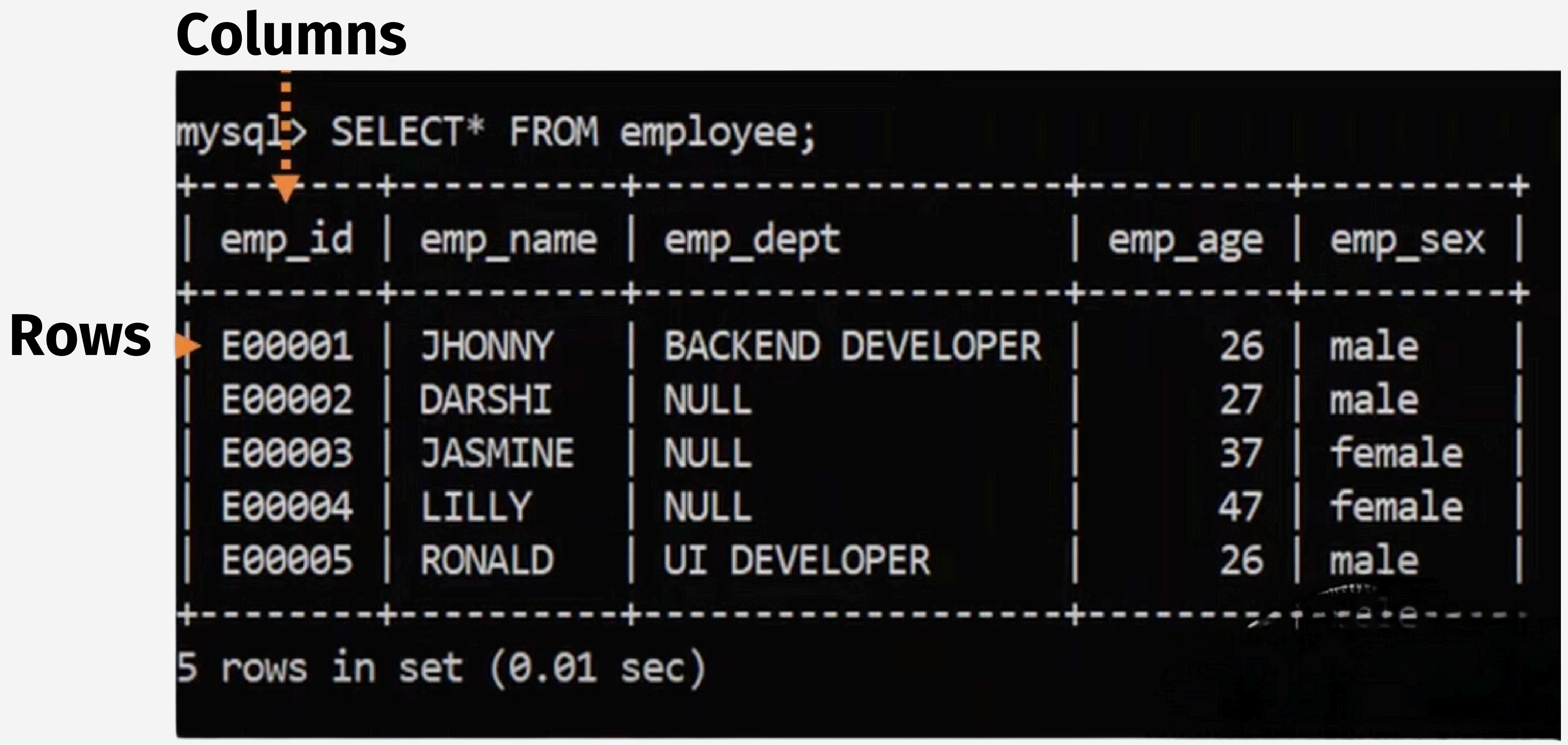
means modifying rows in a table

DELETE

means removing rows from a table

SQL Structure





```
mysql> SELECT* FROM employee;
+--------+----------+--------------------+---------+----------+
| emp_id | emp_name | emp_dept           | emp_age | emp_sex  |
+--------+----------+--------------------+---------+----------+
| E00001 | JHONNY   | BACKEND DEVELOPER  |      26 | male     |
| E00002 | DARSHI   | NULL               |      27 | male     |
| E00003 | JASMINE  | NULL               |      37 | female   |
| E00004 | LILLY    | NULL               |      47 | female   |
| E00005 | RONALD   | UI DEVELOPER       |      26 | male     |
+--------+----------+--------------------+---------+----------+
5 rows in set (0.01 sec)
```

Primary and Foreign Keys

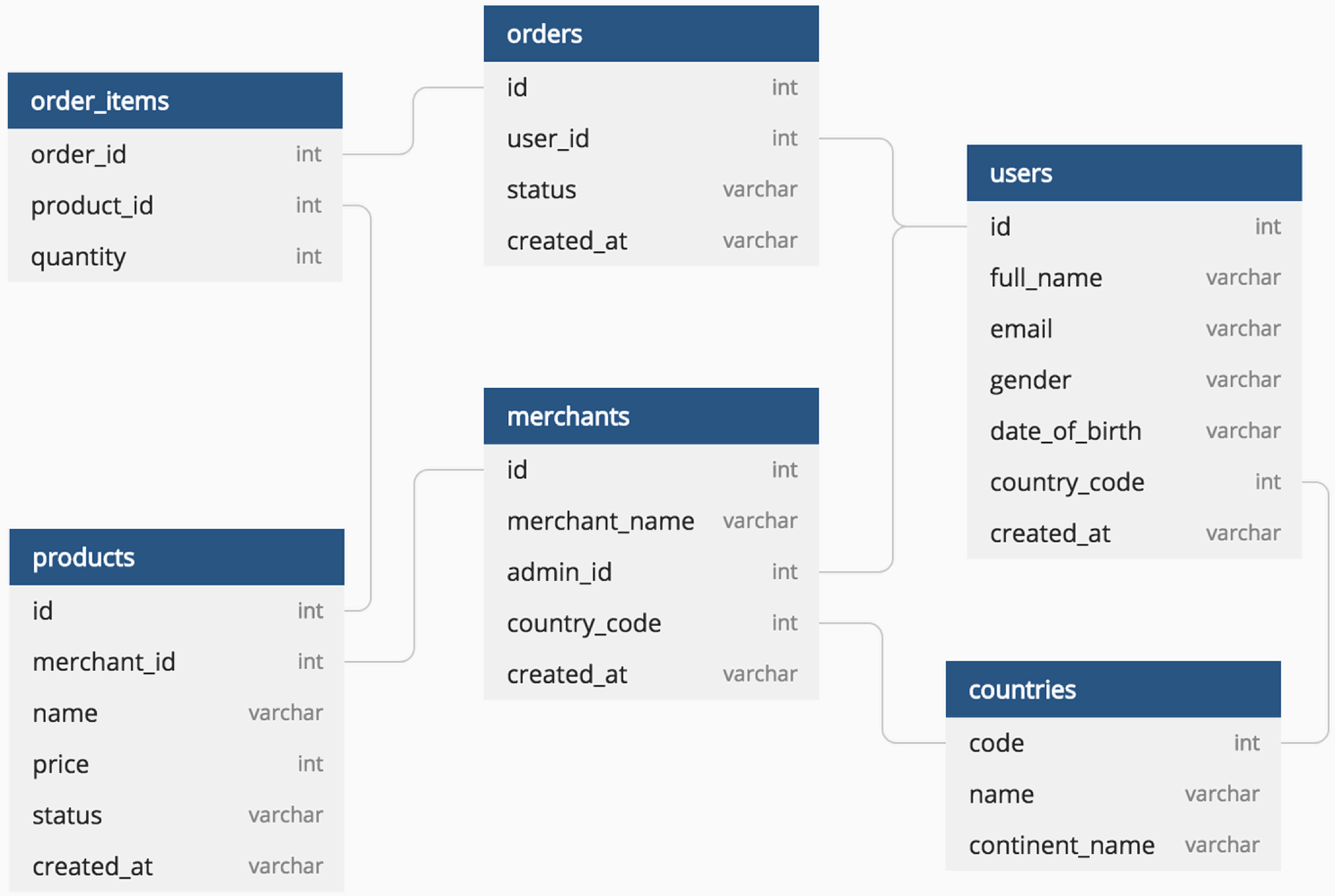
Primary Key (PK)

- A primary key is a unique column we set in a table to easily identify and locate data in queries
- A table can have only one primary key, which should be unique and NOT NULL

Foreign Key (FK)

- A Foreign Key is a column used to link two or more tables together
- A table can have any number of foreign keys and they can be duplicate and NULL values

Database Diagram



orders	
id	int
user_id	int
status	varchar
created_at	varchar

order_items	
order_id	int
product_id	int
quantity	int

users	
id	int
full_name	varchar
email	varchar
gender	varchar
date_of_birth	varchar
country_code	int
created_at	varchar

merchants	
id	int
merchant_name	varchar
admin_id	int
country_code	int
created_at	varchar

products	
id	int
merchant_id	int
name	varchar
price	int
status	varchar
created_at	varchar

countries	
code	int
name	varchar
continent_name	varchar

SELECT Statements

The select statement is used to select data from a database.

```
SELECT column_name FROM table_name;
```

```
SELECT * FROM table_name;
```

```
SELECT DISTINCT column_name FROM table_name;
```

WHERE Statements

The WHERE clause is used to filter the records. It is used to extract only those records that fulfill a specified condition.

```
SELECT column_name FROM table_name
WHERE conditions;
```

Operators in SQL

The SQL reserved words and Characters are called operators, which are used with a WHERE clause in a SQL query

- Arithmetic Operators
  - +, -, *, /, %
- Comparison Operators:
  - =, !=, >, >=
- Logical Operators:
  - ALL, IN, BETWEEN, LIKE, AND, OR, NOT, ANY
- Bitwise Operators:
  - &, |

LIMIT and ORDER BY Clause

The LIMIT clause is used to set an upper limit on the number of rows returned by SQL

```
SELECT column_name FROM table_name
WHERE conditions LIMIT 5;
```

The ORDER BY clause is used to sort the result-set in ascending (ASC) or descending (DESC)

```
SELECT column_name FROM table_name WHERE
conditions ORDER BY column_name ASC
```

String Functions

String Functions are used to perform an operation on input string and return an output string

UPPER()	converts the value of a field to uppercase
LOWER()	converts the value of a field to lowercase
LENGTH()	returns the length of the value in a text field
SUBSTRING()	extracts a substring from a string
NOW()	returns the current system date and time
CONCAT()	adds two or more strings together
REPLACE()	replaces all occurrences of a substring within a string, with a new substring
TRIM()	removes leading or trailing spaces from a string

Aggregate Functions

An aggregate function performs a calculation on multiple values and returns a single value.

Aggregate functions are often used with GROUP BY and SELECT statements

COUNT()	returns number of values
SUM()	returns sum of all values
AVG()	returns average value
MAX()	returns maximum value
MIN()	returns minimum value
ROUND()	rounds a number to a specified number of decimal places

GROUP BY Statement

The GROUP BY statement groups rows that have same values into summary rows
It is often used with aggregate functions (COUNT(), MIN(), MAX(), SUM(), AVG()) to group the result set by one or more columns

```
SELECT column_name(s) FROM table_name
WHERE conditions GROUP BY column_name(s);
```

Example

```
SELECT source,  count(id) FROM
c3_job_application WHERE company_id  = 273
GROUP BY source;
```

HAVING Clause

The HAVING clause is used to apply a filter on the result of GROUP BY based on the specified condition.
The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

```
SELECT column_name(s) FROM table_name
WHERE conditions GROUP BY column_name(s)
HAVING conditions;
```

Example

```
SELECT source,  count(id) FROM
c3_job_application WHERE company_id  = 273
GROUP BY source HAVING count(id) > 100
```

JOIN

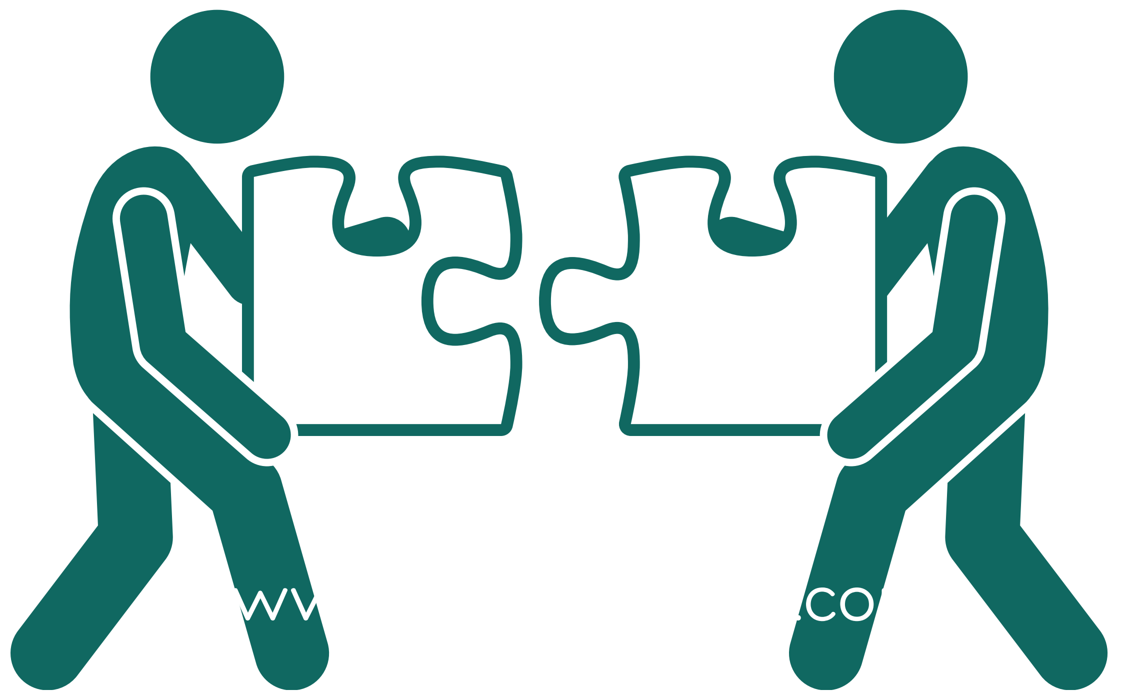
- What is JOIN?
- Use of JOIN
- Types of JOIN
- Which JOIN to use
- JOIN Syntax

SQL JOIN

- **JOIN** means to combine something.
- A **JOIN** clause is used to combine data from two or more tables, based on a related column between them.
- Let’s understand this through an example

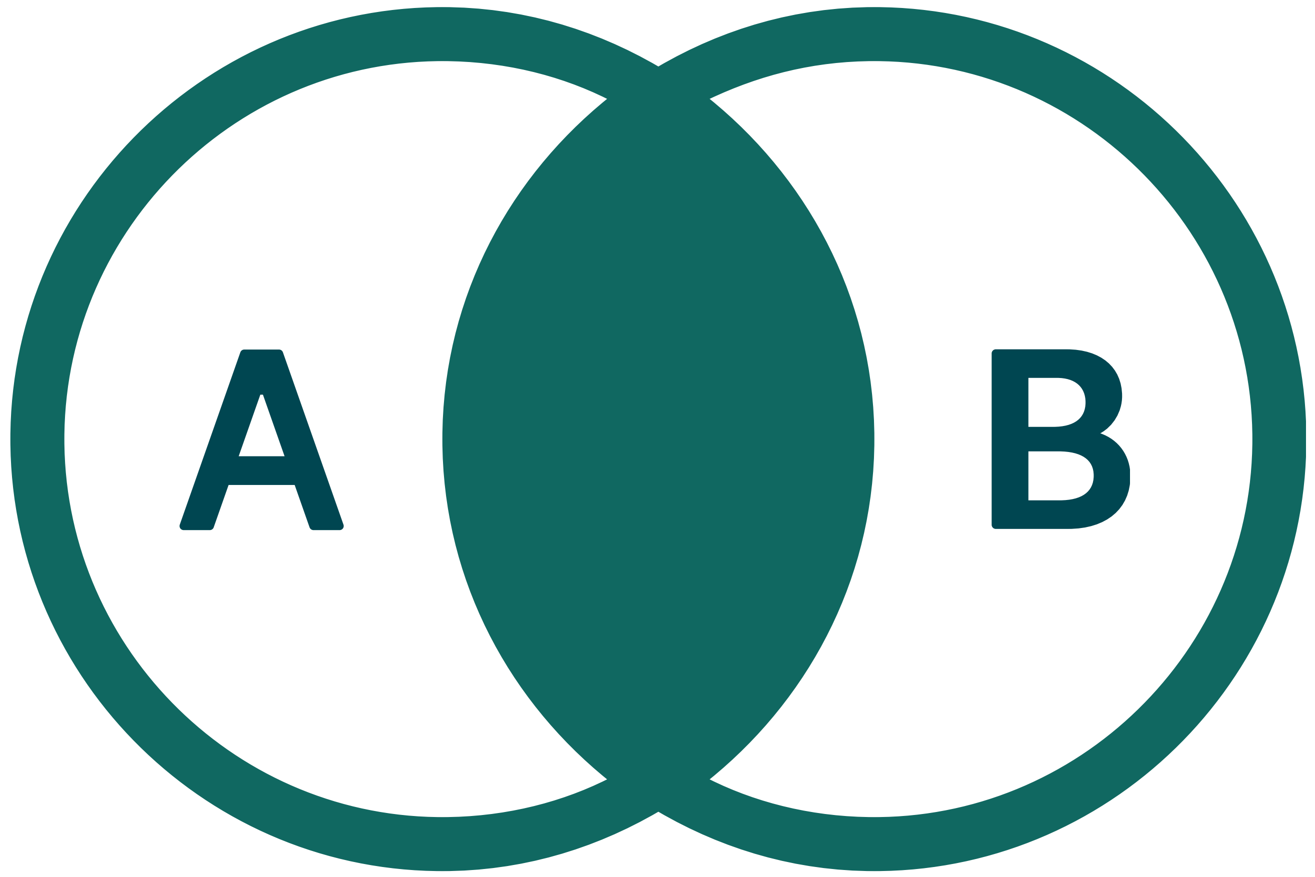
Types of JOIN

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN



INNER JOIN

It returns the records that have matching values in both tables.



INNER JOIN

Syntax

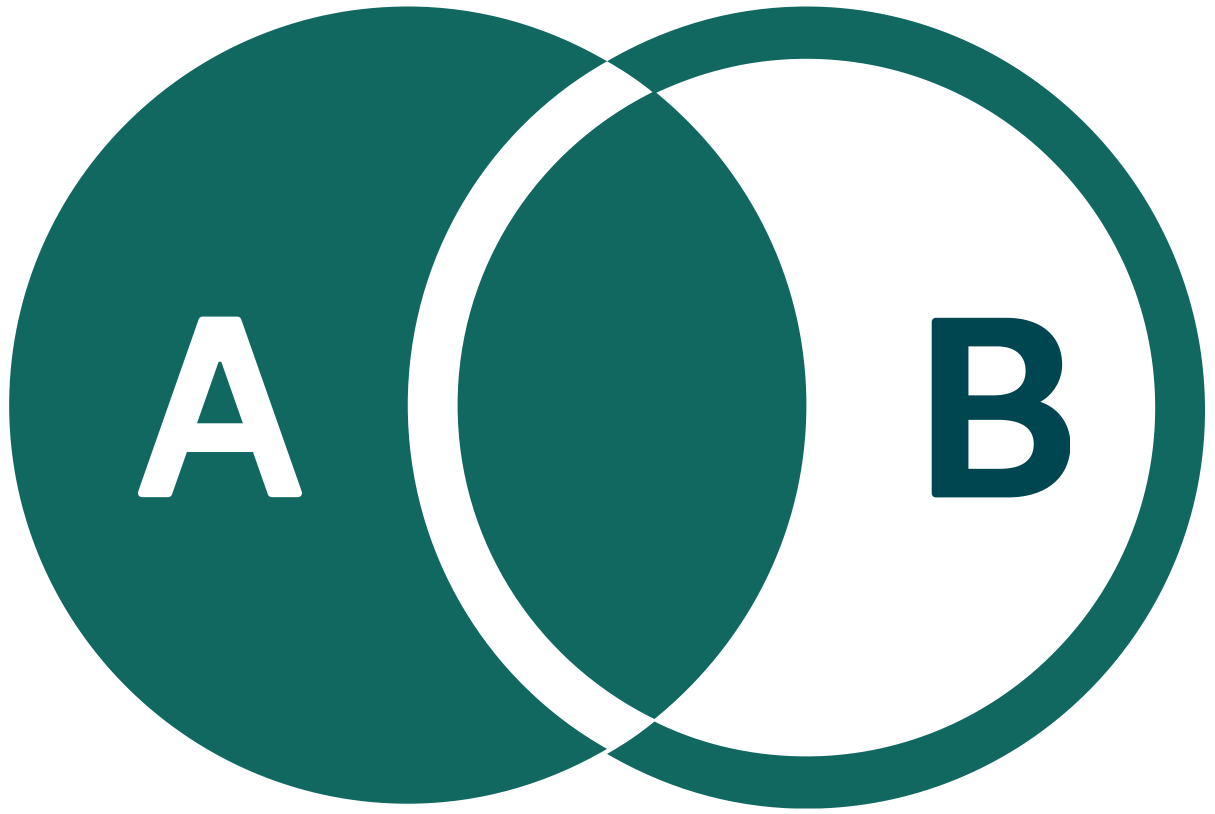
SELECT column_name(s) **FROM** TableA **INNER JOIN** TableB **ON** TableA.column_name = TableB.column_name

Example

SELECT a.id as ‘Application ID’, j.job_title as ‘Job Title’ from c3_job_application a **INNER JOIN** c3_job j **ON** a.job_id = j.id WHERE a.company_id = 273;

LEFT JOIN

It returns all records from the left table, and the matched records from the right table.



LEFT JOIN

Syntax

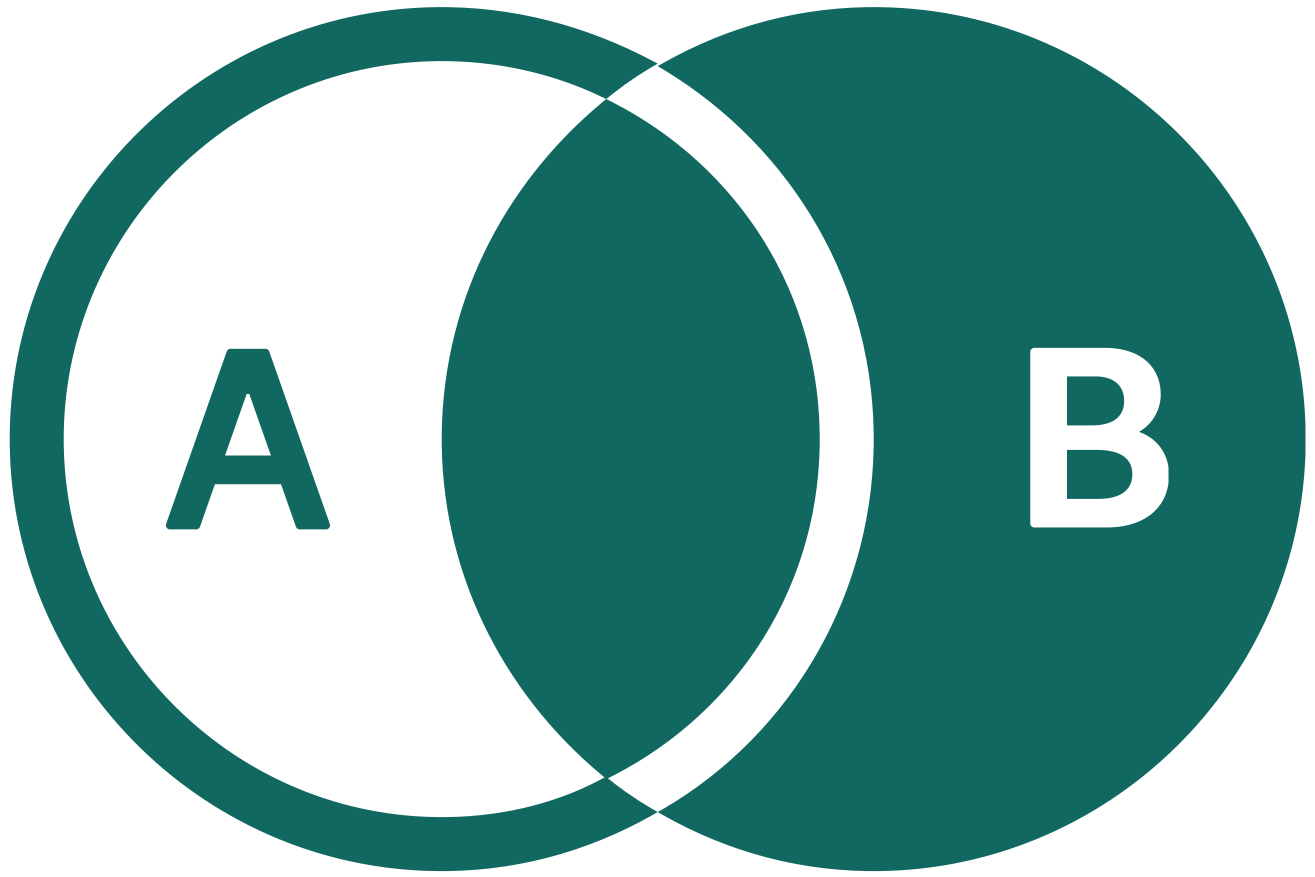
```
SELECT column_name(s) FROM TableA LEFT
JOIN TableB ON TableA.column_name =
TableB.column_name
```

Example

```
SELECT a.id as ‘Application ID’, j.job_title as ‘Job
Title’ from c3_job_application a LEFT JOIN
c3_job j ON a.job_id = j.id WHERE
a.company_id = 273;
```

RIGHT JOIN

It returns all records from the right table, and the matched records from the left table.



RIGHT JOIN

Syntax

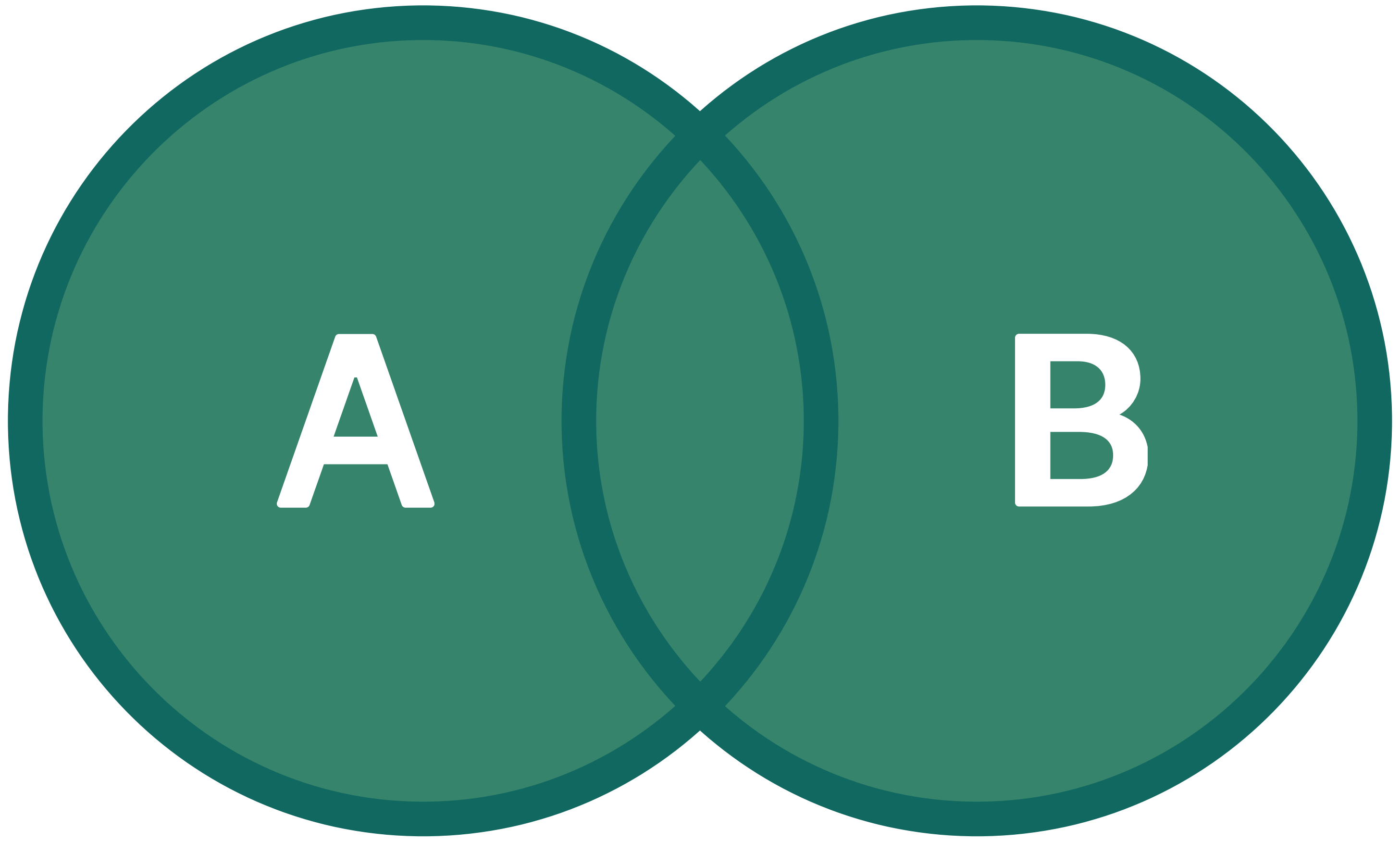
```
SELECT column_name(s) FROM TableA RIGHT
JOIN TableB ON TableA.column_name =
TableB.column_name
```

Example

```
SELECT a.id as ‘Application ID’, j.job_title as ‘Job
Title’ from c3_job_application a RIGHT JOIN
c3_job j ON a.job_id = j.id WHERE
a.company_id = 273;
```

FULL JOIN

It returns all records when there is a match in either left or right table.



FULL JOIN

Syntax

SELECT column_name(s) **FROM** TableA **JOIN** TableB **ON** TableA.column_name = TableB.column_name

Example

SELECT a.id as ‘Application ID’, j.job_title as ‘Job Title’ from c3_job_application a **JOIN** c3_job j **ON** a.job_id = j.id WHERE a.company_id = 273;