# Converting data types

REPORTING IN SQL

SQL

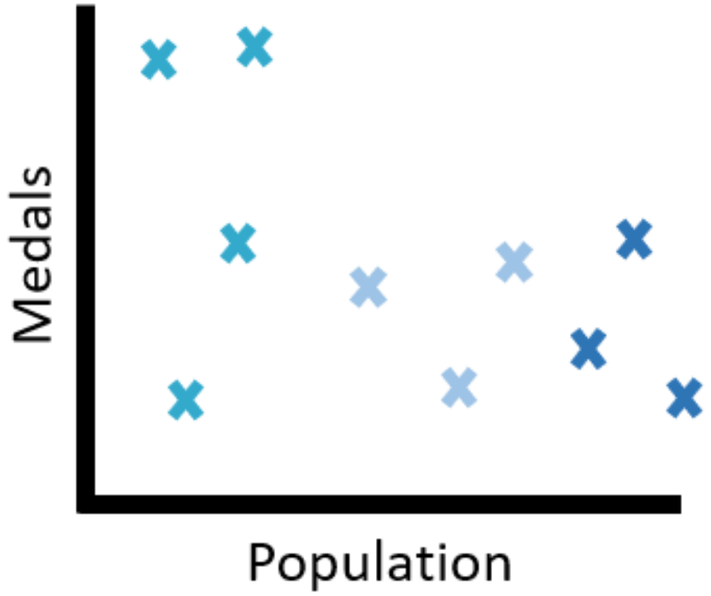
**Tyler Pernes**
Learning & Development Consultant

datacamp

# Chapter goal

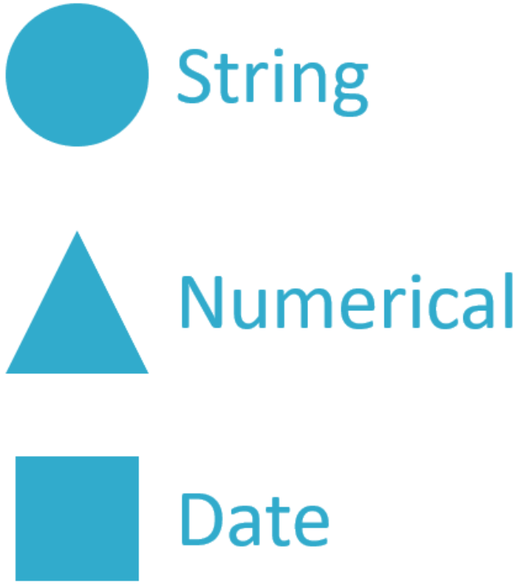
## Medal vs Population Rate

*By country*

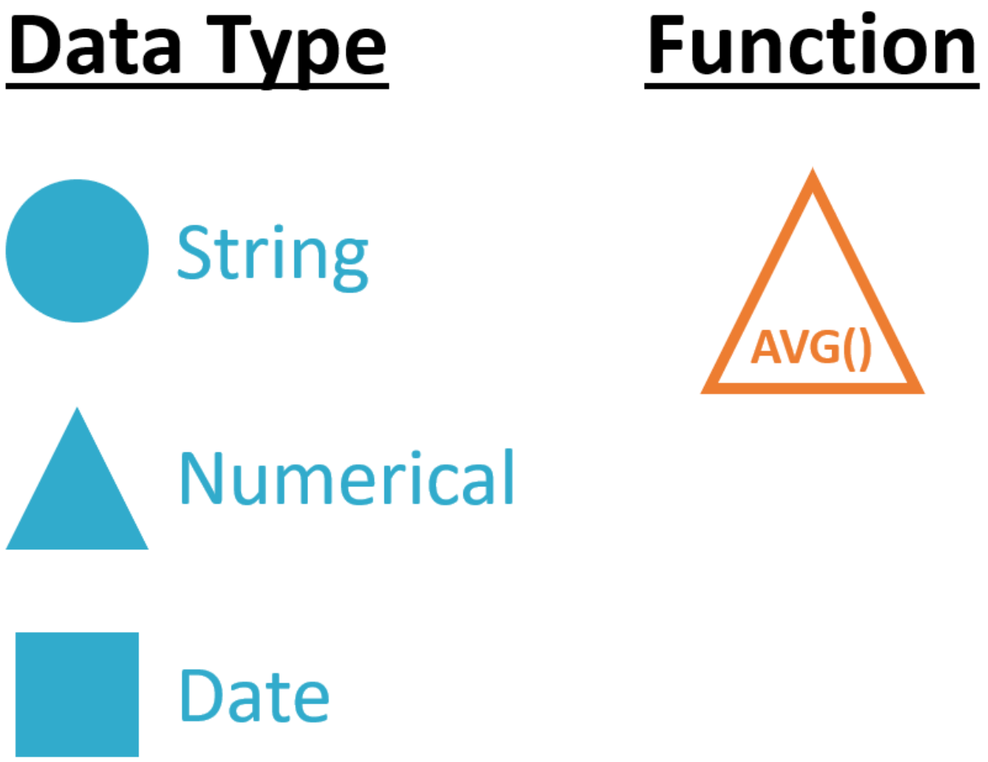


# "Messy" data types

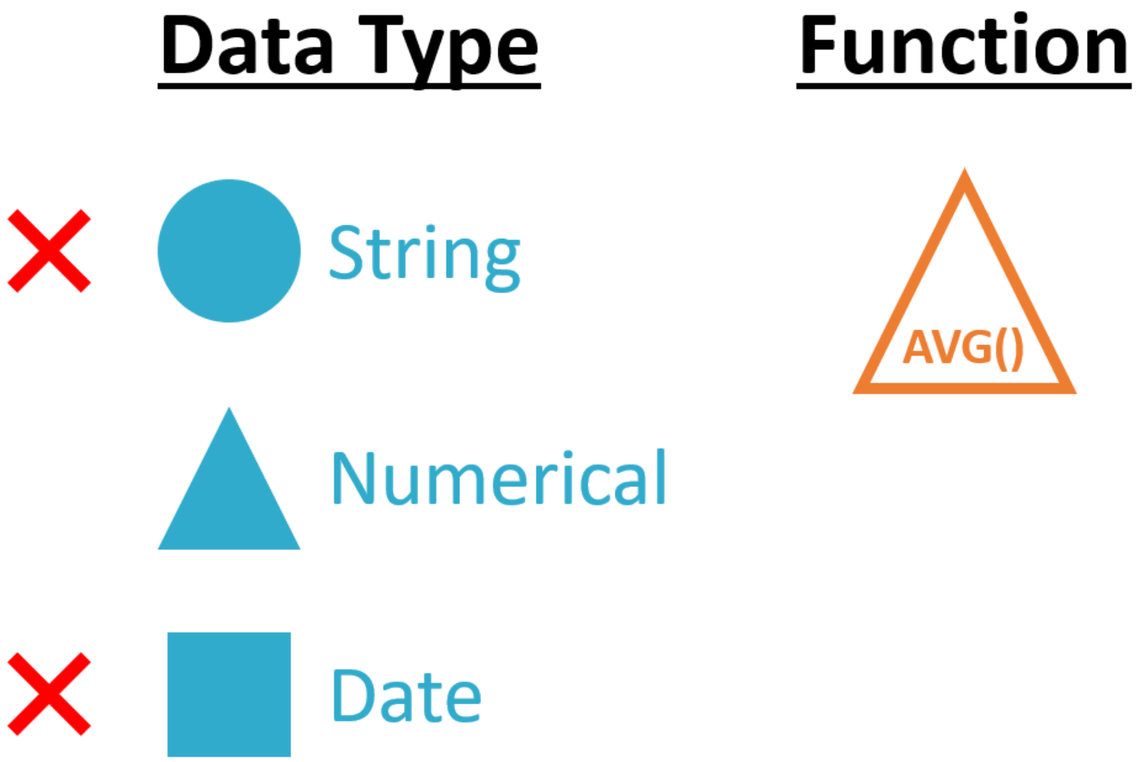
**Data Type**



# Issue 1: Type-specific functions



# Issue 1: Type-specific functions



# Issue 1: Type-specific functions

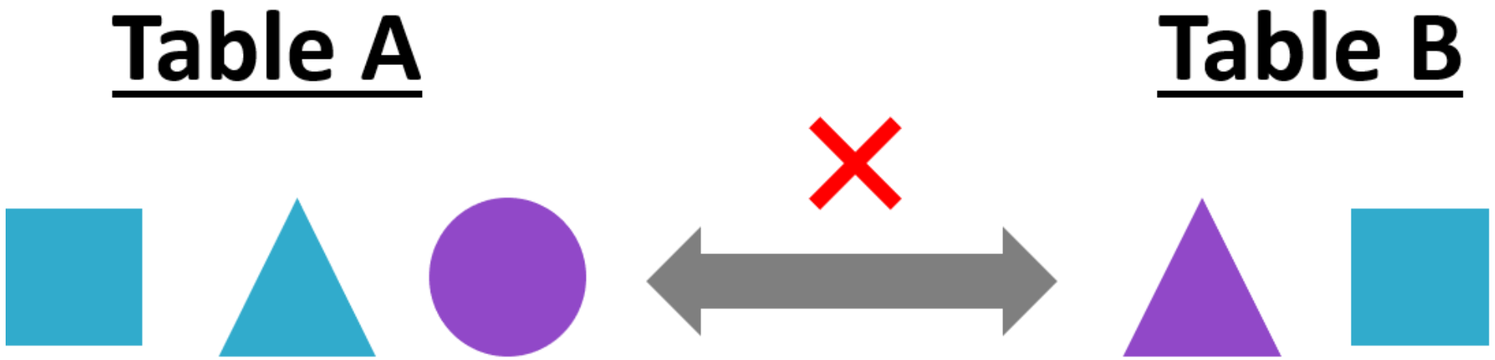
| Data Type | Function |
| --- | --- |
| String | |
| ✓ Numerical | |
| Date | |

# Issue 2: Combining tables (JOIN)

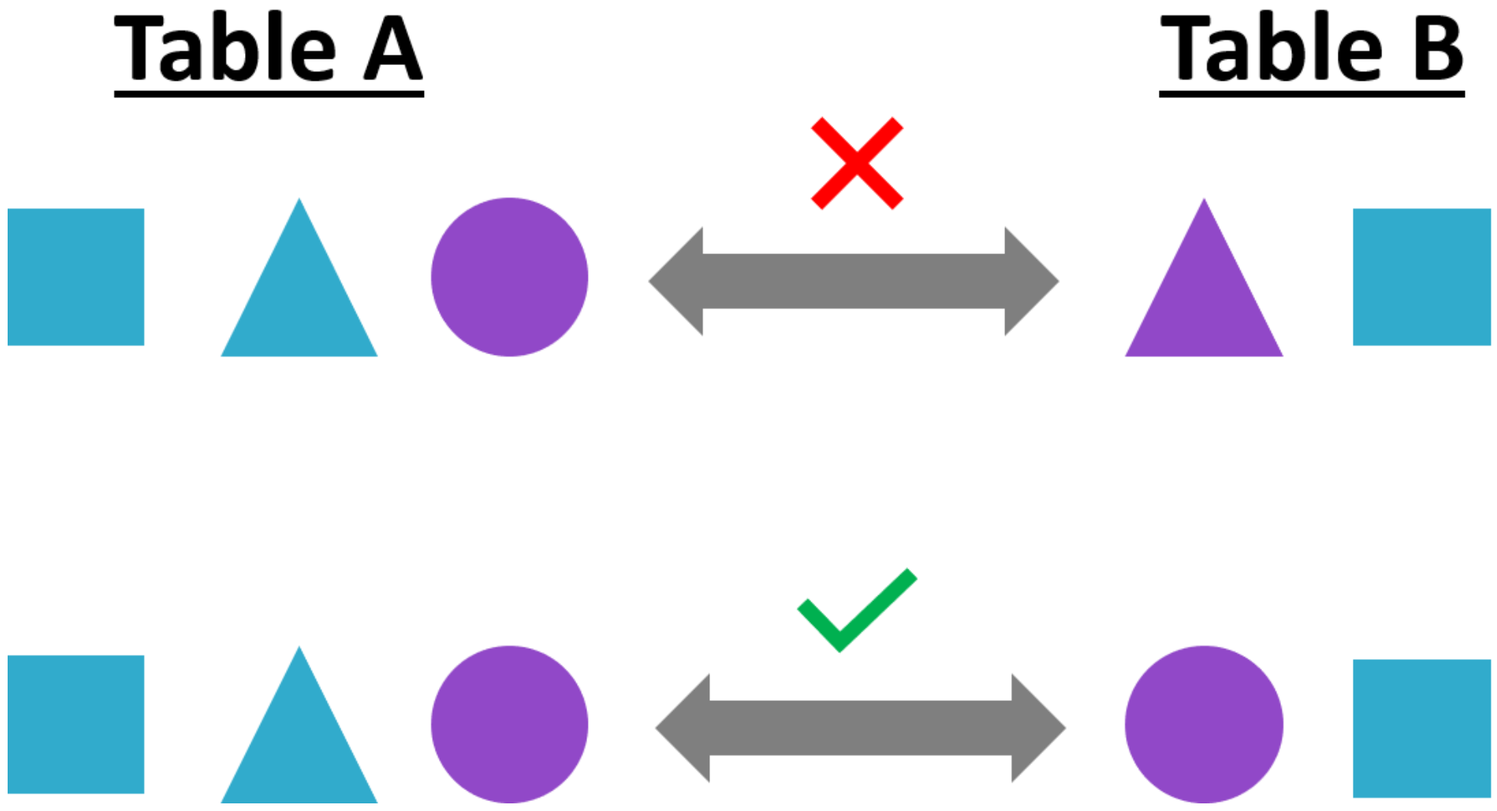
**Table A**

**Table B**

# Issue 2: Combining tables (JOIN)



# Issue 2: Combining tables (JOIN)

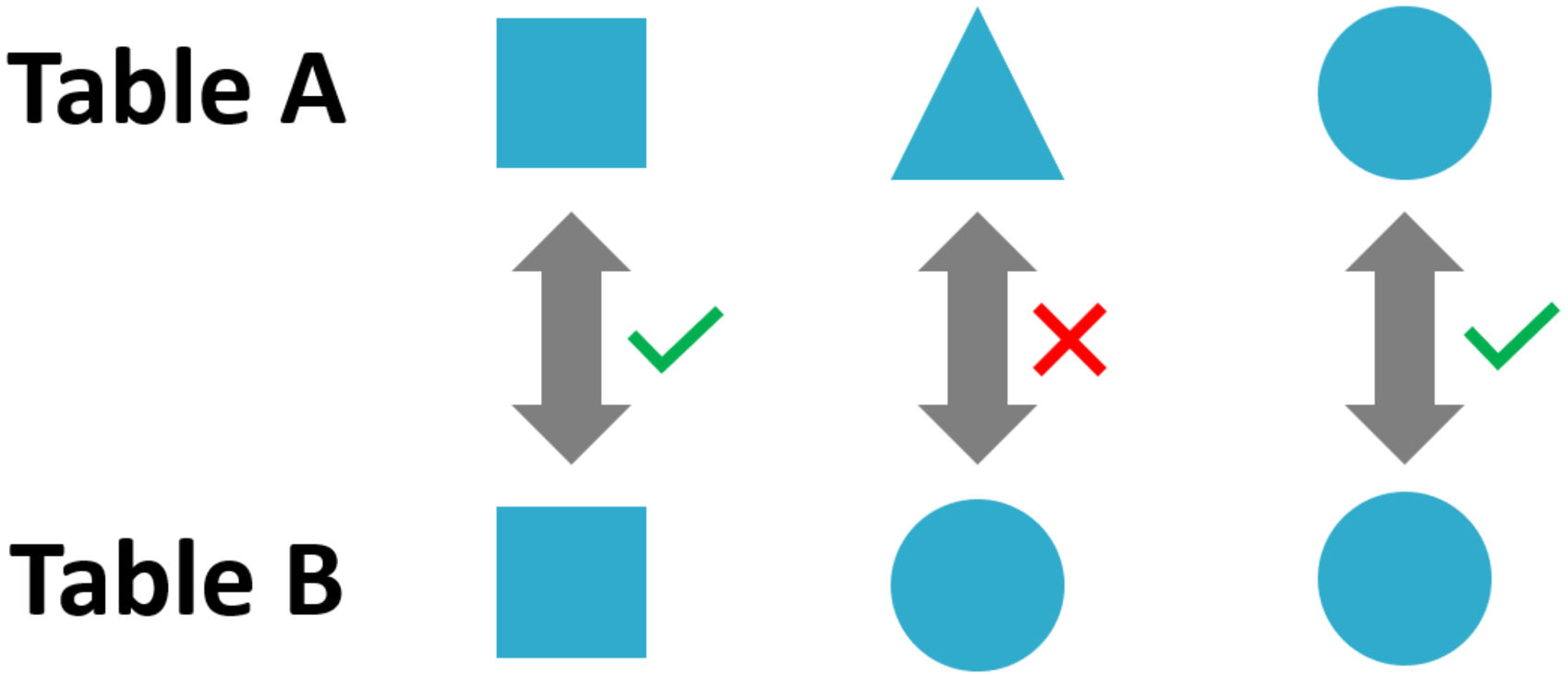


# Issue 2: Combining tables (UNION)

Table A

Table B

# Issue 2: Combining tables (UNION)



# Interpreting errors

## Type-Specific Function Error:

```
SELECT AVG(first_name)
FROM athletes;
```

```
ERROR: Function avg(character varying) does not exist
```

## JOIN Error:

```
SELECT country, continent
FROM countries AS c1
JOIN continents AS c2
ON c1.continent_id = c2.id;
```

```
ERROR: Operator does not exist: integer = character varying
```

# Solution: Wrap it in a CAST()


**Syntax:**

```
CAST(field AS type)
```

**Examples:**

```
CAST(birthday AS date)
CAST(country_id AS int)
```

# CASTing for functions

```
SELECT DATE_PART('month',birthdate)
FROM birthdates;
```

```
ERROR: Can't run DATE_PART on string.
```

```
SELECT DATE_PART('month',
       CAST(birthdate AS date))
FROM birthdates;
```

```
+-----+
| 04  |
| 05  |
+-----+
```

| birthdate |
|---|
| 1994-04-13 |
| 1995-05-16 |

# CASTing for JOINs

```sql
SELECT a.id, b.id
FROM table_a AS a
JOIN table_b AS b
ON a.id = b.id;
```

```
ERROR: Cannot join ON varchar = int.
```

```sql
SELECT a.id, b.id
FROM table_a AS a
JOIN table_b AS b
ON a.id = CAST(b.id AS varchar);
```

```
Query Ran Successfully!
```

**Table A**

VARCHAR

| id |
|---|
| 51 |
| 55 |
| 110 |

**Table B**

INTEGER

| id |
|---|
| 51 |
| 110 |
| 126 |

# Planning for data type issues

- Fix as they come up
- Read error messages!

# Planning for data type issues

```sql
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'countries';
```

```
+---------------+--------------------+
| column_name   | data_type          |
|---------------|--------------------|
| id            | integer            |
| country       | character varying  |
| region        | character varying  |
+---------------+--------------------+
```

# Data type documentation

https://www.postgresql.org/docs/9.5/datatype.html

# Practice time!

REPORTING IN SQL

# Cleaning strings

REPORTING IN SQL

SQL

**Tyler Pernes**
Learning & Development Consultant

datacamp

# Messy strings

```
+----------------------+
| country              |
|----------------------|
| US                   |
| U.S.                 |
| US (United States)   |
| us                   |
|   US                 |
+----------------------+
```

# String functions

| Action | Function(s) |
| --- | --- |

# String functions

| Action | Function(s) |
| --- | --- |
| Remove characters | REPLACE() |
| Replace substrings | REPLACE() |

# Replacing or removing characters

```
+------------+--------+
| country    | points |
|------------|--------|
| US         | 5      |
| U.S.       | 3      |
+------------+--------+
```

```
SELECT REPLACE(country,'.','') AS country_cleaned, SUM(points) as points
FROM original_table
GROUP BY country_cleaned;
```

```
+-----------------+--------+
| country_cleaned | points |
|-----------------|--------|
| US              | 8      |
+-----------------+--------+
```

# String functions

| Action | Function(s) |
| --- | --- |
| Remove characters | REPLACE() |
| Replace substrings | REPLACE() |
| Parse string | LEFT(), RIGHT(), SUBSTRING() |

# Parsing strings

```
+---------------------+--------+
| country             | points |
|---------------------|--------|
| US                  | 5      |
| US (United States)  | 1      |
+---------------------+--------+
```

```sql
SELECT LEFT(country,2) AS country_cleaned, SUM(points) as points
FROM original_table
GROUP BY country_cleaned;
```

```
+------------------+--------+
| country_cleaned  | points |
|------------------|--------|
| US               | 6      |
+------------------+--------+
```

# String functions

| Action | Function(s) |
| --- | --- |
| Remove characters | REPLACE() |
| Replace substrings | REPLACE() |
| Parse string | LEFT(), RIGHT(), SUBSTRING() |
| Change case | UPPER(), LOWER(), INITCAP() |

# Changing case

```
+----------+--------+
| country  | points |
|----------|--------|
| US       | 5      |
| us       | 4      |
+----------+--------+
```

```sql
SELECT UPPER(country) AS country_cleaned, SUM(points) as points
FROM original_table
GROUP BY country_cleaned;
```

```
+------------------+--------+
| country_cleaned  | points |
|------------------|--------|
| US               | 9      |
+------------------+--------+
```

# String functions

| Action | Function(s) |
|---|---|
| Remove characters | REPLACE() |
| Replace substrings | REPLACE() |
| Parse string | LEFT(), RIGHT(), SUBSTRING() |
| Change case | UPPER(), LOWER(), INITCAP() |
| Remove spaces | TRIM() |

# Trimming extra spaces

```
+----------+--------+
| country  | points |
|----------|--------|
| US       | 5      |
|   US     | 2      |
+----------+--------+
```

```sql
SELECT TRIM(country) AS country_cleaned, SUM(points) as points
FROM original_table
GROUP BY country_cleaned;
```

```
+------------------+--------+
| country_cleaned  | points |
|------------------|--------|
| US               | 7      |
+------------------+--------+
```

# Nesting functions

```
original_table
+----------------------+
| country              |
|----------------------|
| US                   |
| U.S.                 |
| US (United States)   |
| us                   |
|   US                 |
+----------------------+
```

```sql
REPLACE(country,'.','')
```

```sql
TRIM(country)
```

```sql
LEFT(country,2)
```

```sql
UPPER(country)
```

# Take it step-by-step

```
REPLACE(country,'.','')
```

```
TRIM(REPLACE(country,'.',''))
```

```
LEFT(TRIM(REPLACE(country,'.','')),2)
```

```
UPPER(LEFT(TRIM(REPLACE(country,'.','')),2))
```

## Final Query:

```
SELECT UPPER(LEFT(TRIM(REPLACE(country,'.','')),2)) AS country_cleaned
FROM original_table
GROUP BY country_cleaned;
```

# Order of nesting matters!

```sql
SELECT TRIM(REPLACE(UPPER(LEFT(country,2)),'.','')) AS country_cleaned
FROM original_table
GROUP BY country_cleaned;
```

```
+-------------------+
| country_cleaned   |
|-------------------|
| US                |
| U                 |
|                   |
+-------------------+
```

# String function documentation

https://www.postgresql.org/docs/9.1/functions-string.html

# Let's practice!

REPORTING IN SQL

# Dealing with nulls

REPORTING IN SQL

SQL

**Tyler Pernes**
Learning & Development Consultant

datacamp

# What does null really mean?

```
+--------+-----------------+
| order  | price_per_unit  |
|--------|-----------------|
| 1      | 4.50            |
| 2      | 2.25            |
| 3      | null            |
+--------+-----------------+
```

- Yet to go through?
- Free?
- Flat price?

# Issues with nulls

```
soccer_games
+---------+-------+-------+
| game_id | home  | away  |
|---------|-------|-------|
| 123     | 3     | 2     |
| 124     | 2     | null  |
| 125     | null  | 1     |
+---------+-------+-------+
```

```sql
SELECT *, home + away AS total_goals
FROM soccer_games;
```

# Issues with nulls

```
+----------+-------+-------+---------------+
| game_id | home  | away  | total_goals  |
|---------|-------|-------|---------------|
| 123     | 3     | 2     | 5             |
| 124     | 2     | null  | null          |
| 125     | null  | 1     | null          |
+----------+-------+-------+---------------+
```

# Issues with nulls

```sql
SELECT
    region,
    COUNT(DISTINCT athlete_id) AS athletes
FROM summer_games AS s
JOIN countries AS c
ON s.country_id = c.id
GROUP BY region;
```

```
+----------+-----------+
| region   | athletes  |
|----------|-----------|
| BALTICS  | 42        |
| OCEANIA  | 62        |
| null     | 10        |
+----------+-----------+
```

- **Unclear** what null represents!

# Fix 1: Filtering nulls

```
original_table
+--------+-----------------+
| order  | price_per_unit  |
|--------|-----------------|
| 1      | 4.50            |
| 2      | 2.25            |
| 3      | null            |
+--------+-----------------+
```

```sql
SELECT *
FROM original_table
WHERE price_per_unit IS NOT NULL;
```

# Fix 1: Filtering nulls

```
+---------+-----------------+
| order   | price_per_unit  |
|---------|-----------------|
| 1       | 4.50            |
| 2       | 2.25            |
+---------+-----------------+
```

# Fix 2: COALESCE()

Syntax: `COALESCE(field, null_replacement)`

```sql
SELECT
    COALESCE(region,'Independent Athletes') AS region,
    COUNT(DISTINCT athlete_id) AS athletes
FROM summer_games AS s
JOIN countries AS c
ON s.country_id = c.id;
```

```
%20----------------------%20-----------%20
| region                | athletes  |
|----------------------|-----------|
| BALTICS               | 42        |
| OCEANIA               | 62        |
| Independent Athletes  | 10        |
%20----------------------%20-----------%20
```

# Fix 2: COALESCE()

```
soccer_games
+---------+-------+-------+
| game_id | home  | away  |
|---------|-------|-------|
| 123     | 3     | 2     |
| 124     | 2     | null  |
| 125     | null  | 1     |
+---------+-------+-------+
```

```sql
SELECT *, COALESCE(home,0) + COALESCE(away,0) AS total_goals
FROM soccer_games;
```

# Fix 2: COALESCE()

```
+----------+-------+-------+---------------+
| game_id | home  | away  | total_goals  |
|---------|-------|-------|---------------|
| 123     | 3     | 2     | 5             |
| 124     | 2     | null  | 2             |
| 125     | null  | 1     | 1             |
+----------+-------+-------+---------------+
```

# Nulls as a result of a query

**Causes:**

- `LEFT JOIN` does not match all rows
- No `CASE` statement conditional is satisfied
- Several others!

# Measuring the impact of nulls

Ratio of **rows** that are null

```
SELECT SUM(CASE when country IS NULL then 1 else 0 end) / SUM(1.00)
FROM orders;
```

```
+-------+
| .12   |
+-------+
```

Ratio of **revenue** that is null

```
SELECT SUM(CASE when country IS NULL then revenue else 0 end) / SUM(revenue)
FROM orders;
```

```
+-------+
| .25   |
+-------+
```

# Practice time!

REPORTING IN SQL

# Report duplication

REPORTING IN SQL

SQL

**Tyler Pernes**
Learning & Development Consultant

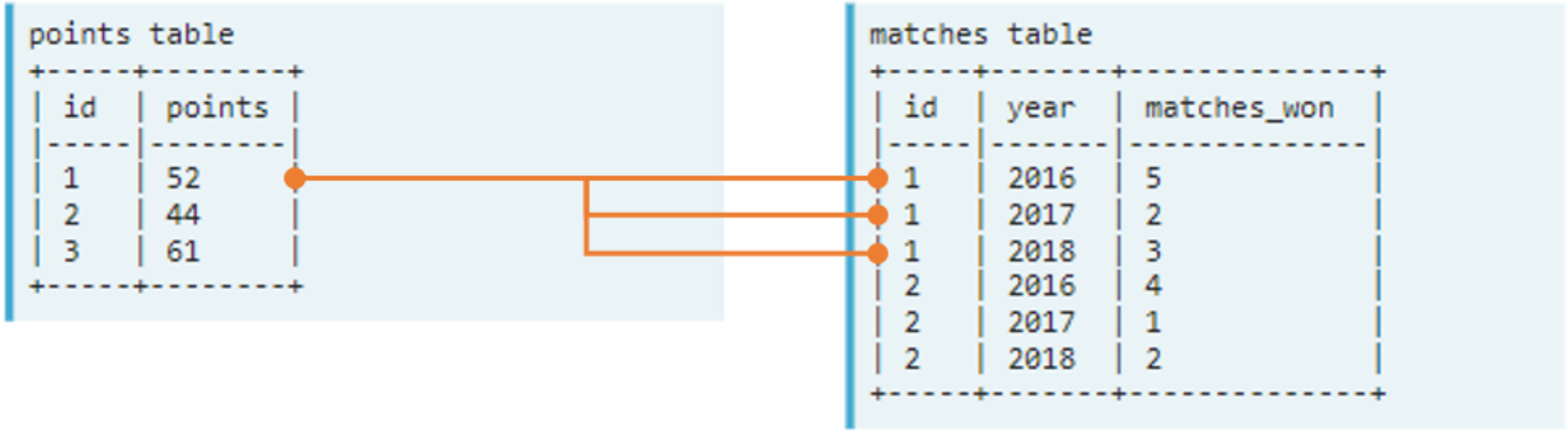
datacamp

# What causes duplication?

```
points table
+-----+--------+
| id  | points |
|-----|--------|
| 1   | 52     |
| 2   | 44     |
| 3   | 61     |
+-----+--------+
```

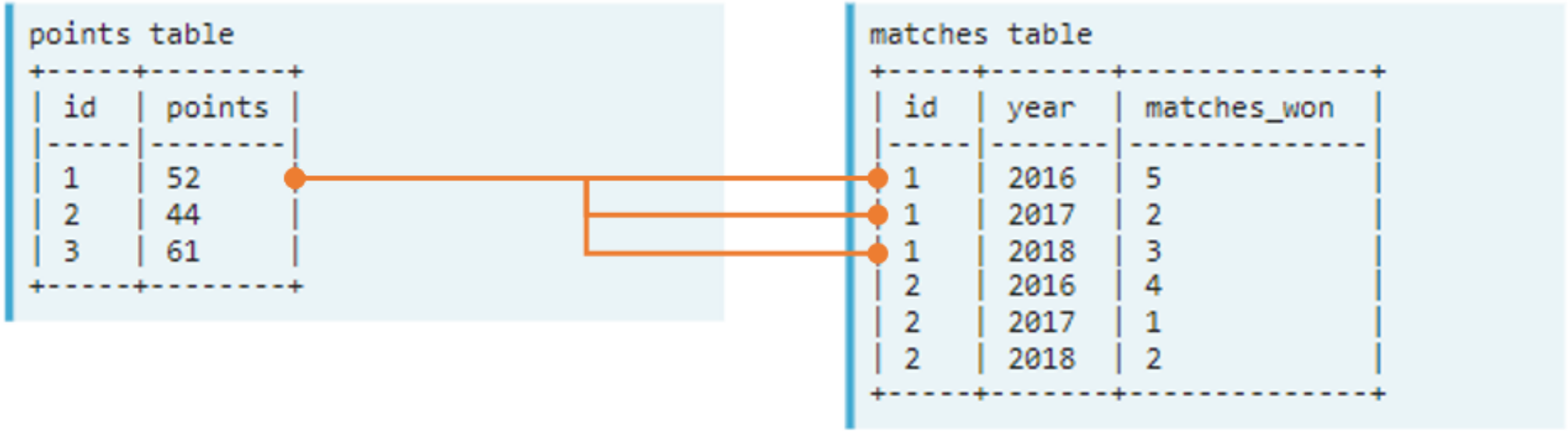
```
matches table
+-----+-------+--------------+
| id  | year  | matches_won  |
|-----|-------|--------------|
| 1   | 2016  | 5            |
| 1   | 2017  | 2            |
| 1   | 2018  | 3            |
| 2   | 2016  | 4            |
| 2   | 2017  | 1            |
| 2   | 2018  | 2            |
+-----+-------+--------------+
```

# What causes duplication?




```sql
SELECT p.id, SUM(points) AS points, SUM(matches_win) AS matches_won
FROM points AS p
JOIN matches AS m ON p.id = m.id
GROUP BY p.id;
```

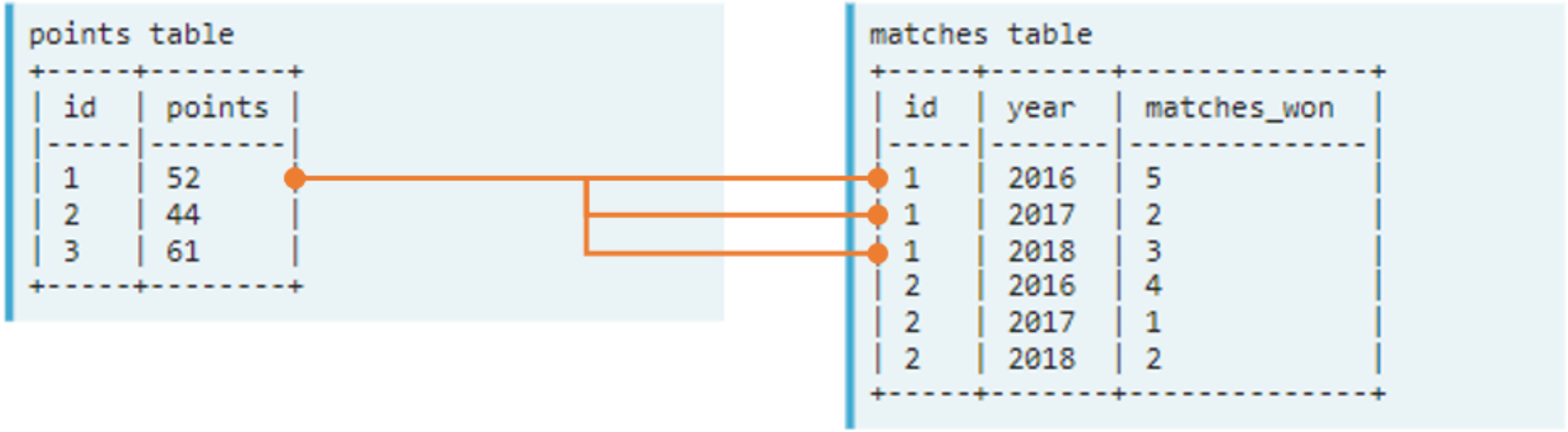
# What causes duplication?




```
+-----+--------+--------------+
| id  | points | matches_won  |
|-----|--------|--------------|
| 1   | 156    | 10           |
+-----+--------+--------------+
```

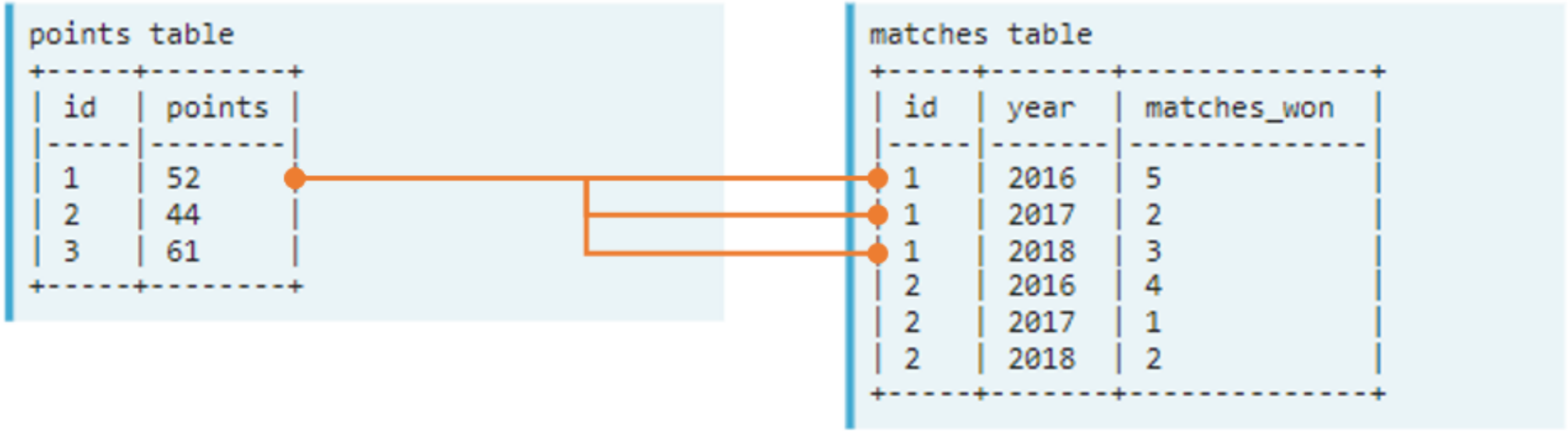
[1] will result in a points value three times what it should be, in this case, 156.

# What causes duplication?




```
Intermediate Table
+-----+------+--------------+---------+
| id  | year | matches_won  | points  |
|-----|------|--------------|---------|
| 1   | 2016 | 5            | 52      |
| 1   | 2017 | 2            | 52      |
| 1   | 2018 | 3            | 52      |
+-----+------+--------------+---------+
```

# What causes duplication?




```
Intermediate Table
+-----+------+--------------+---------+
| id  | year | matches_won  | points  |
|-----|------|--------------|---------|
| 1   | 2016 | 5            | 52      |<--
| 1   | 2017 | 2            | 52      |<-- SUM(points) = 52 x 3 = 156
| 1   | 2018 | 3            | 52      |<--
+-----+------+--------------+---------+
```

# Ways to fix duplication

## 1. Remove aggregations

```sql
SELECT p.id, points, SUM(matches_won) AS matches_won
FROM points AS p
JOIN matches AS m ON p.id = m.id
GROUP BY p.id, points;
```

```
+-----+--------+--------------+
| id  | points | matches_won  |
|-----|--------|--------------|
| 1   | 52     | 10           |
+-----+--------+--------------+
```

# Ways to fix duplication

```
points table
+-----+-------+--------+
| id  | year  | points |
|-----|-------|--------|
| 1   | 2016  | 30     |
| 1   | 2017  | 22     |
| 2   | 2016  | 20     |
| 2   | 2017  | 24     |
+-----+-------+--------+
```

```
matches table
+-----+-------+--------------+
| id  | year  | matches_win  |
|-----|-------|--------------|
| 1   | 2016  | 5            |
| 1   | 2017  | 2            |
| 1   | 2018  | 3            |
| 2   | 2016  | 4            |
| 2   | 2017  | 1            |
| 2   | 2018  | 2            |
+-----+-------+--------------+
```

# Ways to fix duplication

points table

| id | year | points |
|---|---|---|
| 1 | 2016 | 30 |
| 1 | 2017 | 22 |
| 2 | 2016 | 20 |
| 2 | 2017 | 24 |

matches table

| id | year | matches_win |
|---|---|---|
| 1 | 2016 | 5 |
| 1 | 2017 | 2 |
| 1 | 2018 | 3 |
| 2 | 2016 | 4 |
| 2 | 2017 | 1 |
| 2 | 2018 | 2 |

## 2. Add field to JOIN statement

```
SELECT p.id, SUM(points) AS points, SUM(matches_win) AS matches_won
FROM points AS p
JOIN matches AS m ON p.id = m.id AND p.year = m.year
GROUP BY p.id;
```

# Ways to fix duplication

points table

| id | year | points |
|---|---|---|
| 1 | 2016 | 30 |
| 1 | 2017 | 22 |
| 2 | 2016 | 20 |
| 2 | 2017 | 24 |

matches table

| id | year | matches_win |
|---|---|---|
| 1 | 2016 | 5 |
| 1 | 2017 | 2 |
| 1 | 2018 | 3 |
| 2 | 2016 | 4 |
| 2 | 2017 | 1 |
| 2 | 2018 | 2 |

## 2. Add field to JOIN statement

```
SELECT p.id, SUM(points) AS points, SUM(matches_win) AS matches_won
FROM points AS p
JOIN matches AS m ON p.id = m.id AND p.year = m.year
GROUP BY p.id;
```

# Ways to fix duplication

```sql
SELECT id, SUM(matches_won)
FROM matches
GROUP BY id;
```

```
+-----+--------------+
| id  | matches_won  |
|-----|--------------|
| 1   | 10           |
| 2   | 7            |
+-----+--------------+
```

# Ways to fix duplication

```
points table
+-----+--------+
| id  | points |
|-----|--------|
| 1   | 52     |
| 2   | 44     |
| 3   | 61     |
+-----+--------+
```

```
matches subquery
+-----+--------------+
| id  | matches_won  |
|-----|--------------|
| 1   | 10           |
| 2   | 7            |
+-----+--------------+
```

## 3. Rollup using subquery

```sql
SELECT p.id, points, matches_won
FROM points AS p
JOIN
    (SELECT id, SUM(matches_won) AS matches_won
    FROM matches
    GROUP BY id) AS m
ON p.id = m.id;
```

# Ways to fix duplication

1. Remove aggregations
2. Add field to JOIN statement
3. Rollup using subquery

# Identifying duplication

**Value in original table:**

```sql
SELECT SUM(points) AS total_points
FROM points;
```

```
total_points = 52
```

**Value in query:**

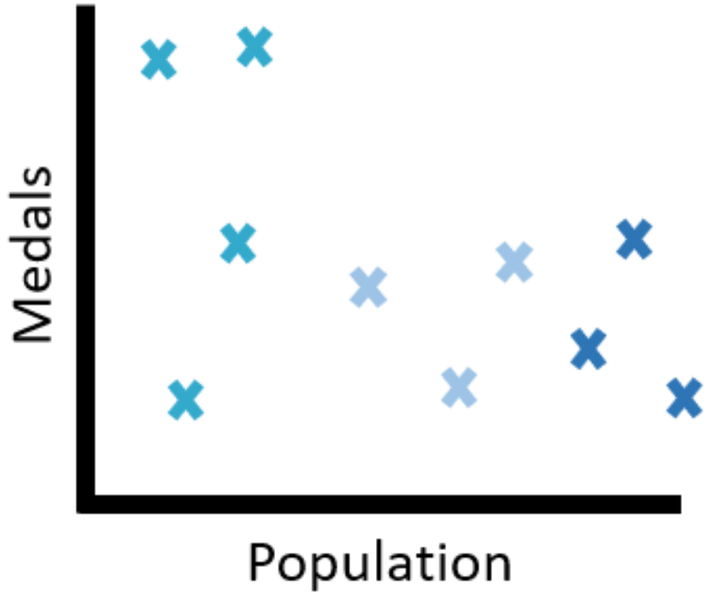
```sql
SELECT SUM(points) AS total_points
FROM
    (SELECT p.id, SUM(points) AS points
    FROM points AS p
    JOIN matches AS m ON p.id = m.id
    GROUP BY p.id) AS subquery;
```

```
total_points = 156
```

# Chapter goal

## Medal vs Population Rate

*By country*



# Practice time!

REPORTING IN SQL