FLASHBACK: SO FAR WE HAVE SEEN QUERIES AS STANDALONE COMMANDS THAT FETCH DATA FROM A DATABASE

BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)

BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

## BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

## BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

# BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

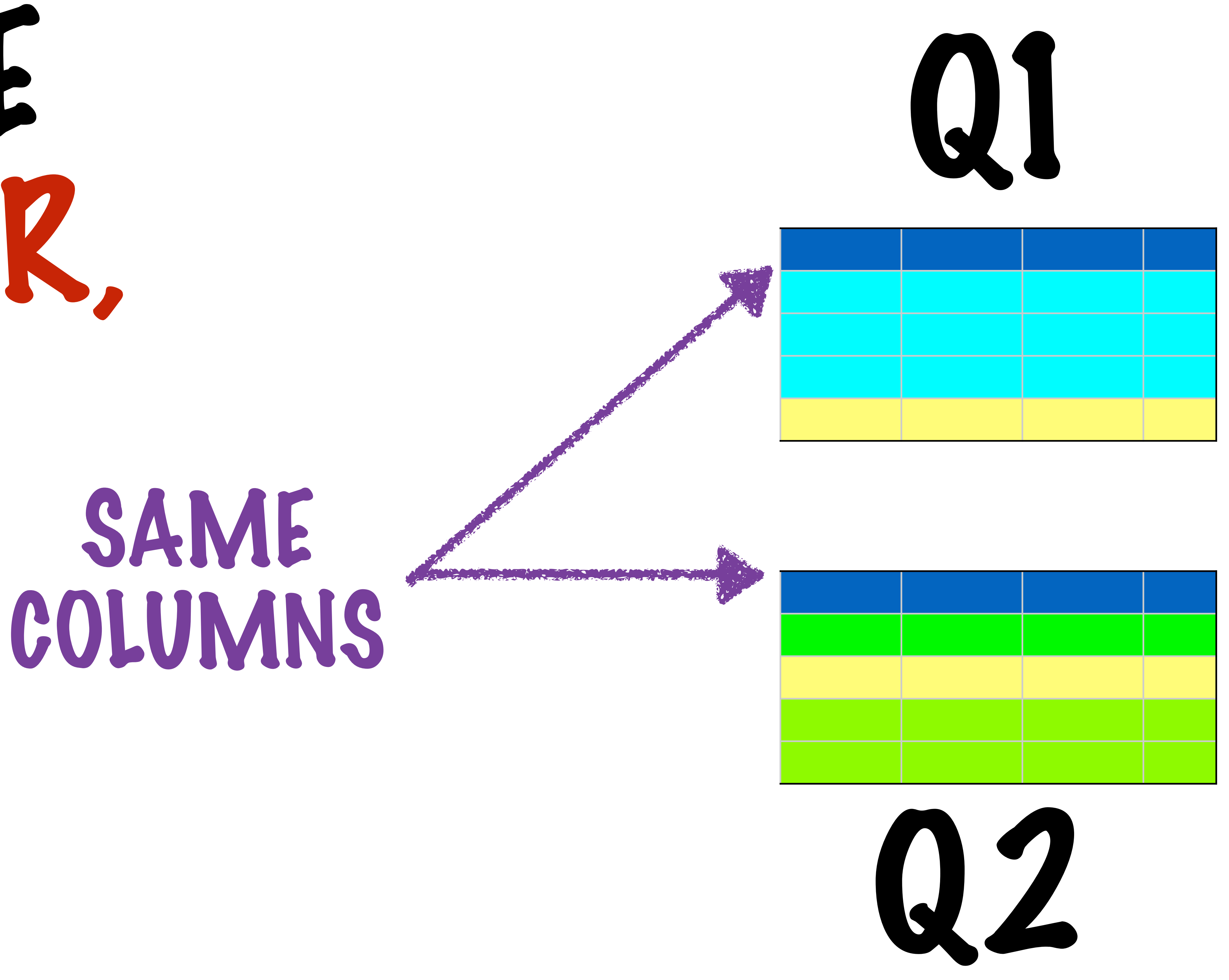
WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

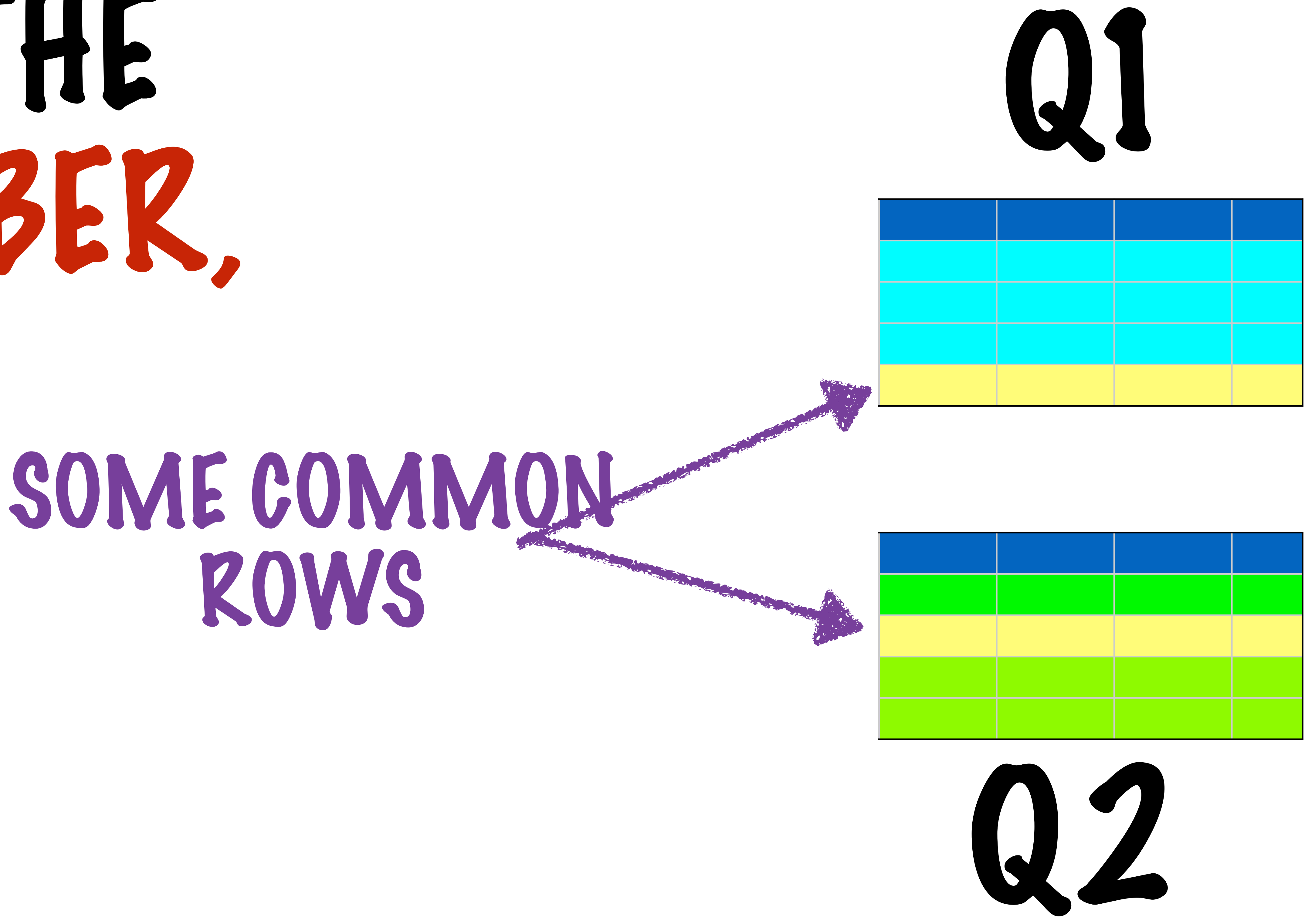
WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)



## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

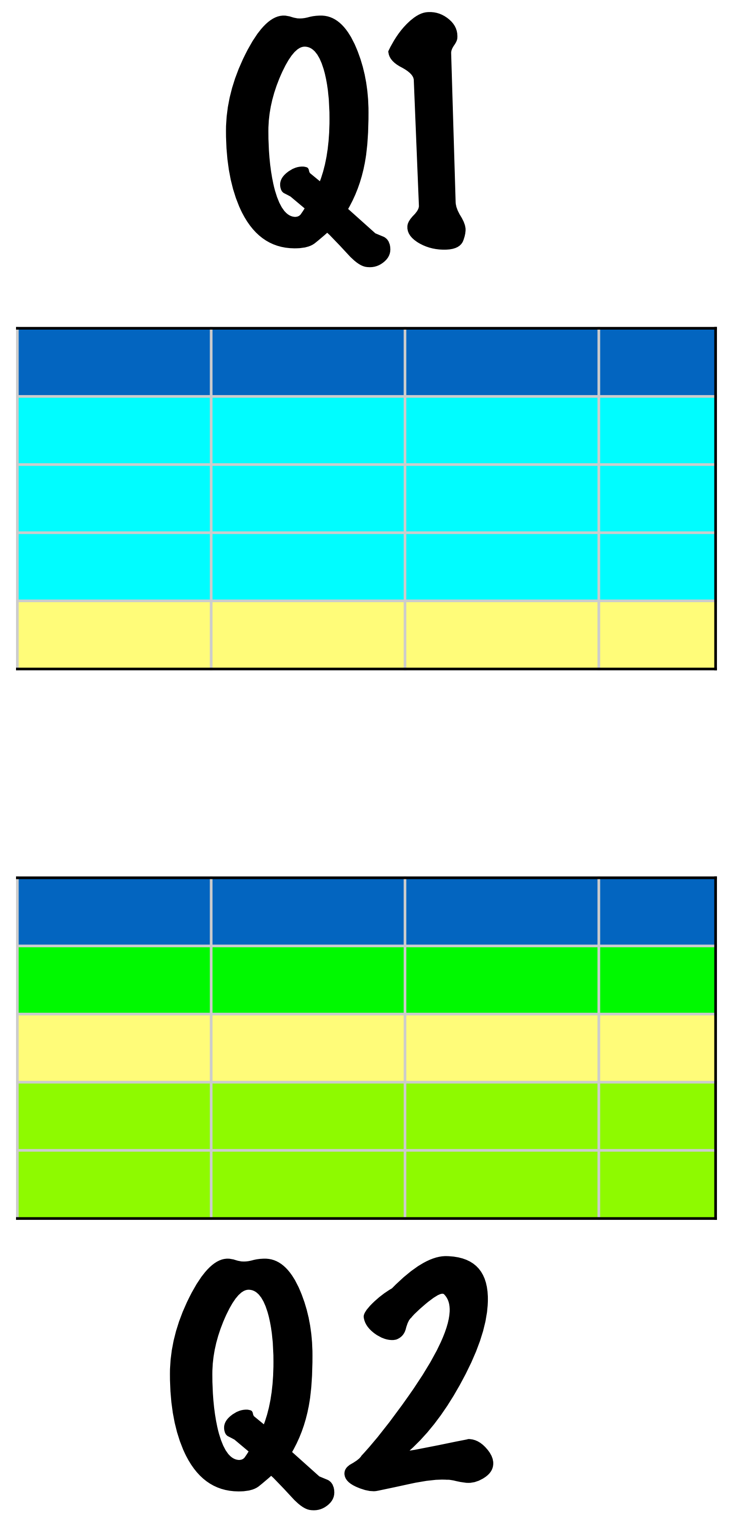
PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)



## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)

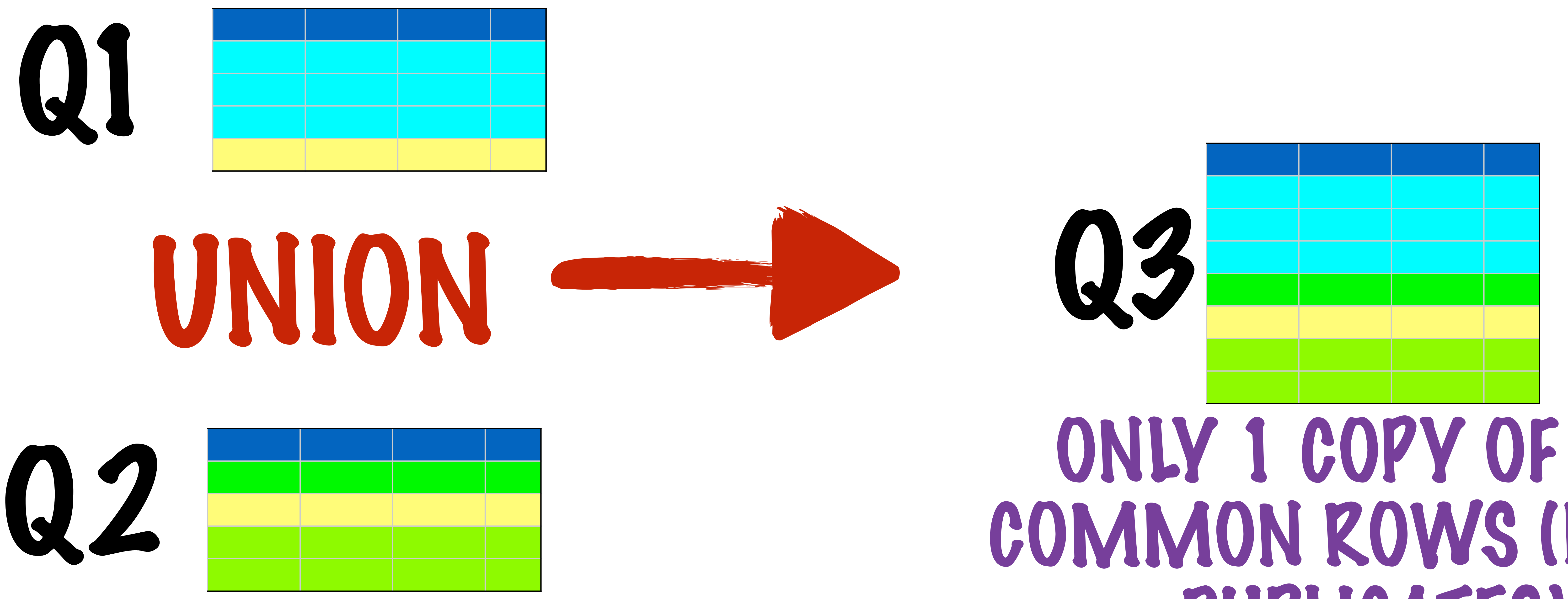
A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)

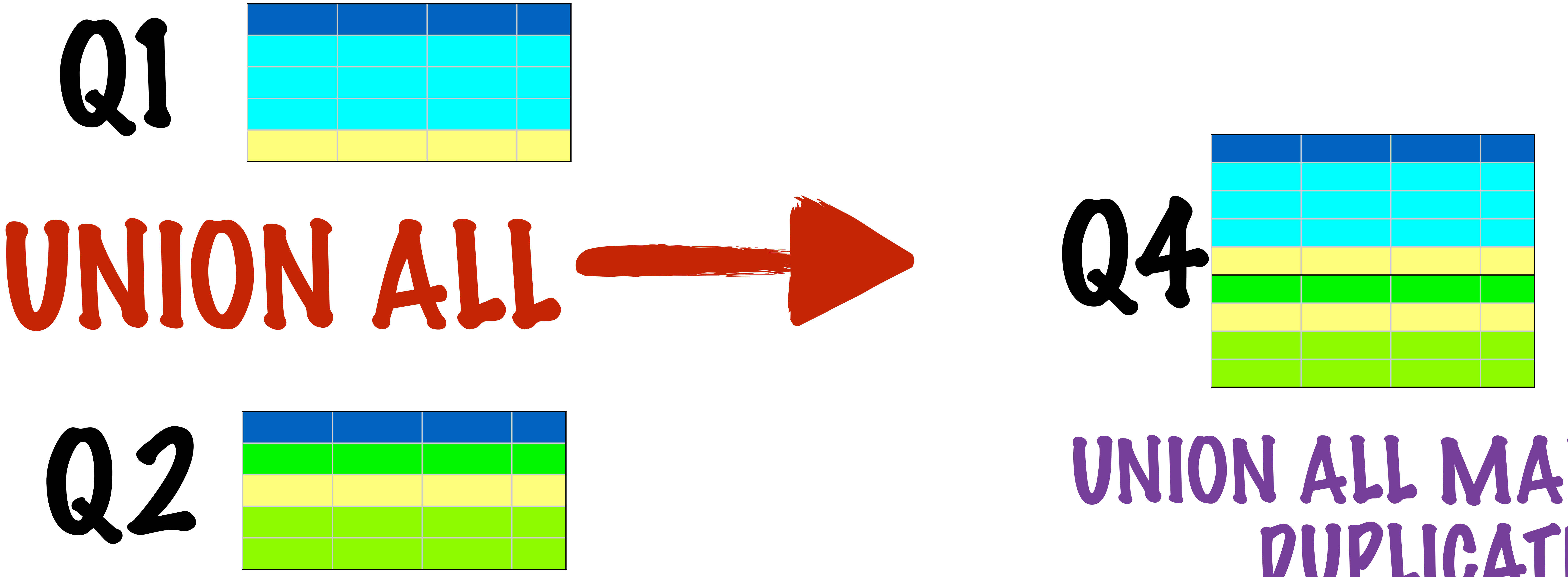


ONLY 1 COPY OF THE COMMON ROWS (I.E. NO DUPLICATES)

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)

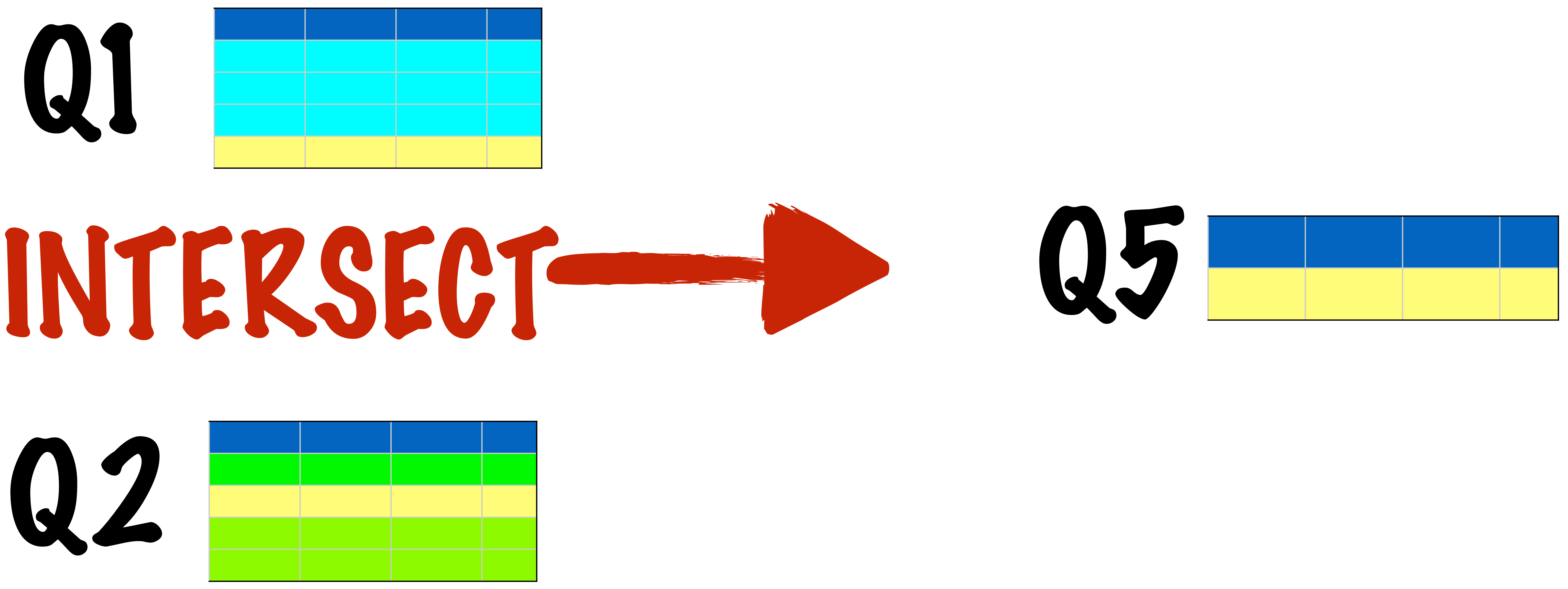


UNION ALL MAINTAINS DUPLICATES

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)

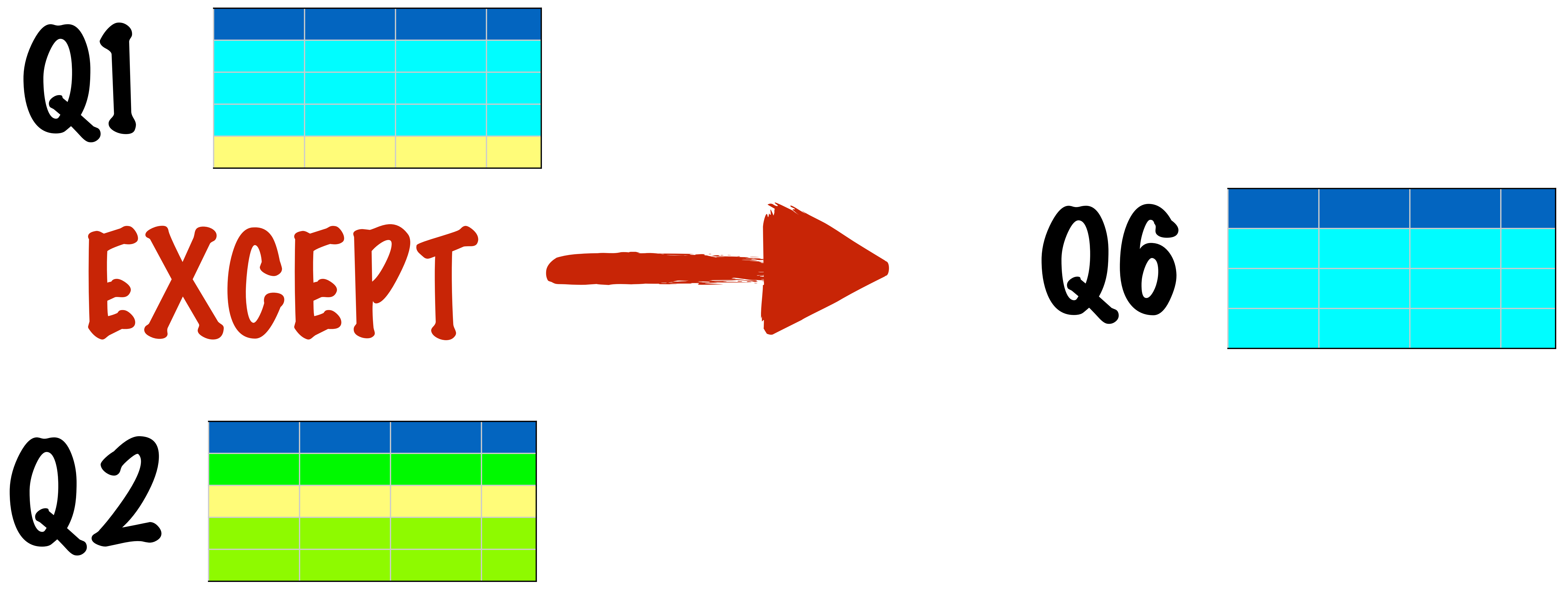
A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)

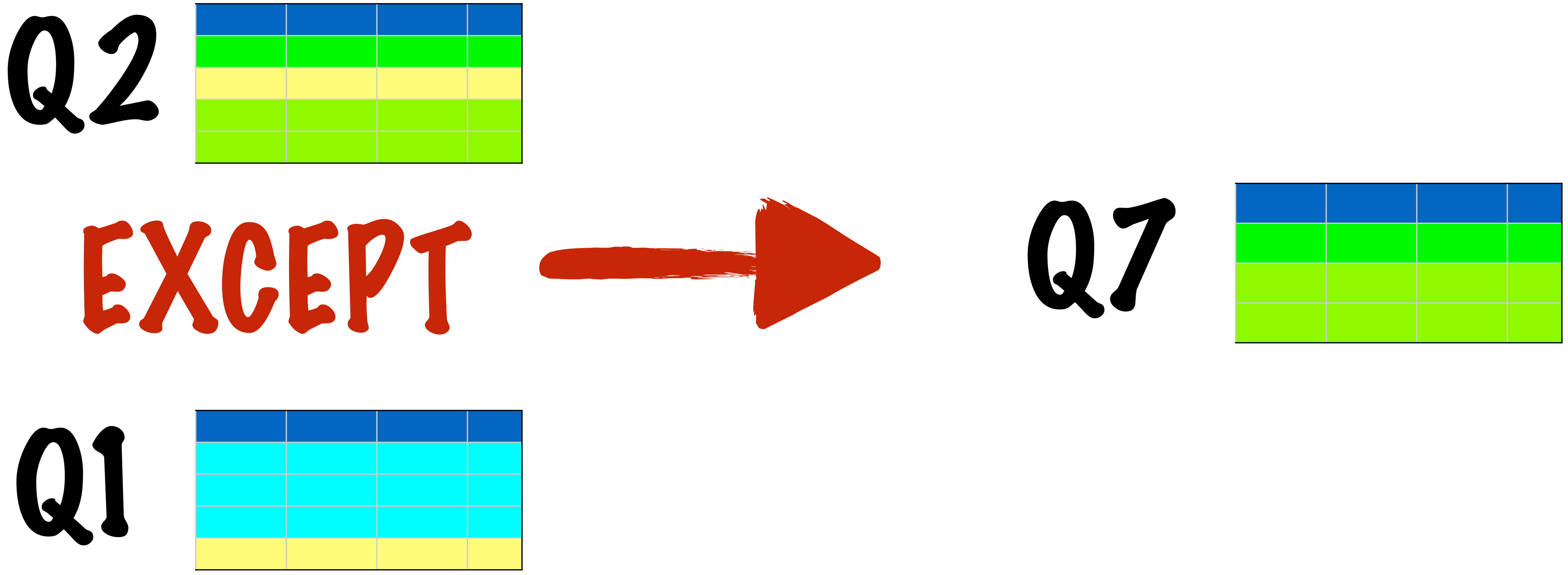
A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

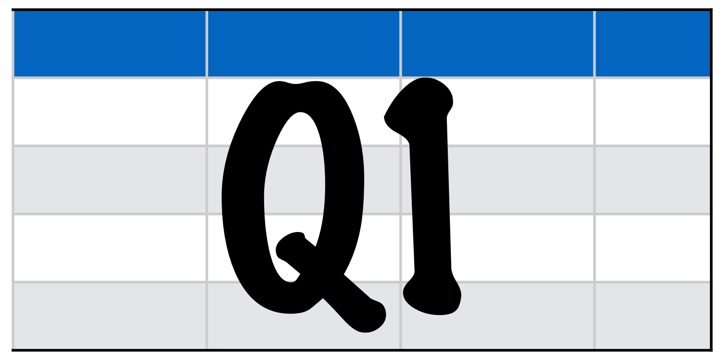
PROVIDED THEY HAVE THE SAME COLUMNS (NUMBER, ORDER AND TYPE)

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

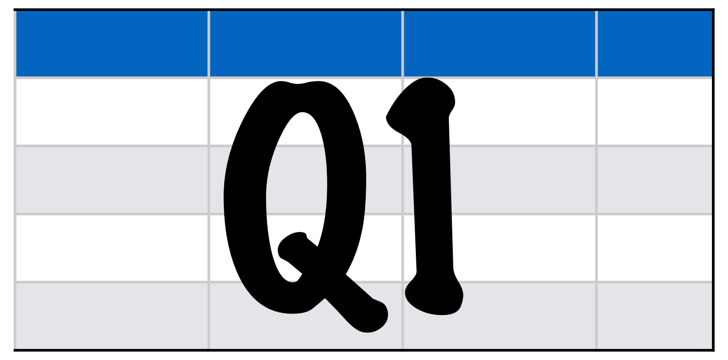
A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



NOW, UNION, INTERSECTION AND DIFFERENCE ARE SET OPERATIONS

## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)

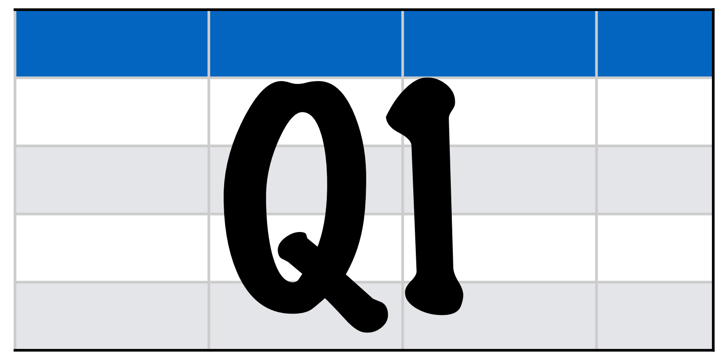


NOW, UNION, INTERSECTION AND DIFFERENCE ARE SET OPERATIONS

E-R THEORY TELLS THAT A TABLE IS A BAG OF TUPLES

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



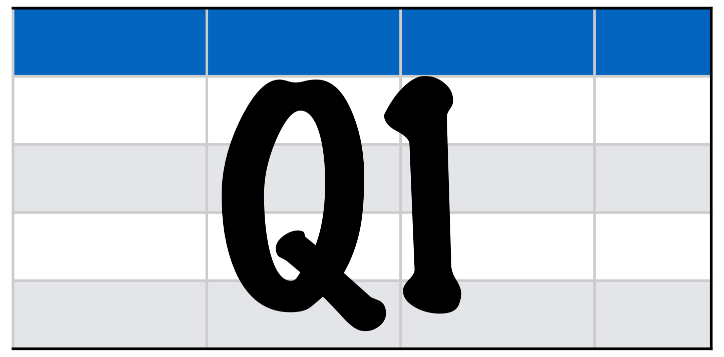
NOW, UNION, INTERSECTION AND DIFFERENCE ARE SET OPERATIONS

E-R THEORY TELLS THAT A TABLE IS A BAG OF TUPLES

A SET CAN'T CONTAIN DUPLICATES, BUT A BAG CAN

## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



NOW, UNION, INTERSECTION AND DIFFERENCE ARE SET OPERATIONS

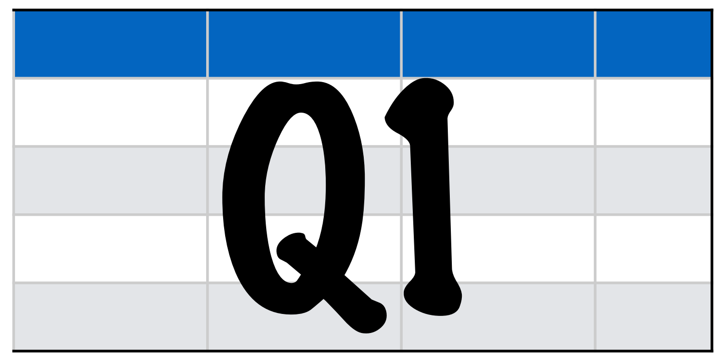
E-R THEORY TELLS THAT A TABLE IS A BAG OF TUPLES

A SET CAN'T CONTAIN DUPLICATES, BUT A BAG CAN

SO, BY DEFAULT, UNION WILL ELIMINATE DUPLICATES

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

A QUERY IS A COMMAND THAT RETURNS A TABLE (ROWS AND COLUMNS)



NOW, UNION, INTERSECTION AND DIFFERENCE ARE SET OPERATIONS

A SET CAN'T CONTAIN DUPLICATES, BUT A BAG CAN

E-R THEORY TELLS THAT A TABLE IS A BAG OF TUPLES

SO, BY DEFAULT, UNION WILL ELIMINATE DUPLICATES

BUT SQL HAS UNION ALL TO KEEP DUPES

WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

NOW, UNION, INTERSECTION AND DIFFERENCE ARE SET OPERATIONS

SO, BY DEFAULT, UNION WILL ELIMINATE DUPLICATES

BUT SQL HAS UNION ALL TO KEEP DUPES

ALSO THE INDIVIDUAL QUERIES IN A UNION CAN NOT USE ORDER BY ← ELEMENTS OF A SET ARE NOT ORDERED

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PET OWNERS

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |

SAME COLUMNS (NUMER, ORDER AND TYPE - NAMES COULD DIFFER)

APT OWNERS

| FlatNumber | Name |
|---|---|
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

PET OWNERS

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |

1 COMMON ROW

APT OWNERS

| FlatNumber | Name |
|---|---|
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

```
(SELECT APTNUMBER,
NAME FROM PetOwners)
```

**UNION**

```
(SELECT FLATNUMBER AS
APTNUMBER, NAME FROM
AptOwners);
```

PET OWNERS

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |

APT OWNERS

| FlatNumber | Name |
|---|---|
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

ONLY 1 COPY OF THE COMMON ROWS (I.E. NO DUPLICATES)

```
(SELECT APTNUMBER,
NAME FROM PetOwners)
```

**UNION**

```
(SELECT FLATNUMBER AS
APTNUMBER, NAME FROM
AptOwners);
```

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

```
(SELECT APTNUMBER,
NAME FROM PetOwners)
```

**UNION ALL**

```
(SELECT FLATNUMBER AS
APTNUMBER, NAME FROM
AptOwners);
```

PET OWNERS

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |

APT OWNERS

| FlatNumber | Name |
|---|---|
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

## WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

```
(SELECT APTNUMBER,
NAME FROM PetOwners)

UNION ALL

(SELECT FLATNUMBER AS
APTNUMBER, NAME FROM
AptOwners);
```

UNION ALL MAINTAINS DUPLICATES

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

```
(SELECT APTNUMBER, NAME FROM
  PetOwners ORDER BY NAME)
```

**UNION**

**WON'T WORK!!**

```
(SELECT FLATNUMBER AS
APTNUMBER, NAME FROM
AptOwners ORDER BY NAME);
```

PET OWNERS

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |

APT OWNERS

| FlatNumber | Name |
|---|---|
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

# WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

## THIS IS FINE!

```
((SELECT
APTNUMBER, NAME
FROM PetOwners)
UNION
(SELECT FLATNUMBER AS
APTNUMBER, NAME FROM
AptOwners)) Order by
APTNUMBER;
```

PET OWNERS

| AptNumber | Name |
|---|---|
| 123 | John |
| 345 | Tim |
| 349 | Nikhil |
| 567 | Bilal |

APT OWNERS

| FlatNumber | Name |
|---|---|
| 234 | Mary |
| 567 | Bilal |
| 897 | Alan |
| 903 | Ellen |

# BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT

# BUT IN REALITY, QUERIES ARE PRETTY PLUG-AND-PLAY

WE COULD USE ONE QUERY INSIDE ANOTHER (VIA SUBQUERIES)

✓ WE COULD CALCULATE THE UNION, INTERSECTION OR DIFFERENCE OF 2 QUERIES

WE COULD USE A SUBQUERY TO POPULATE A TABLE VIA INSERT