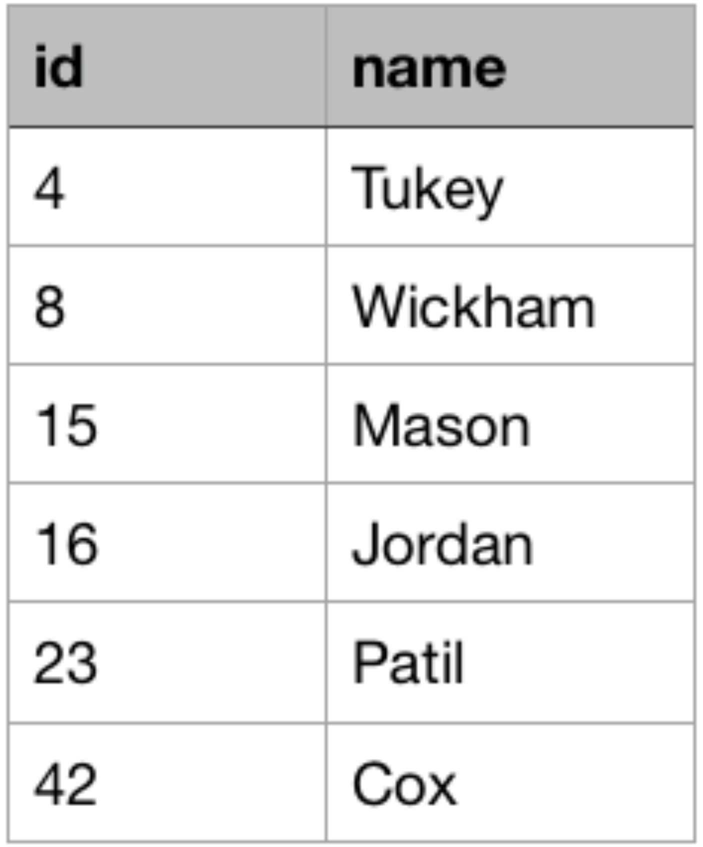
Joining Data Tables

Motivation

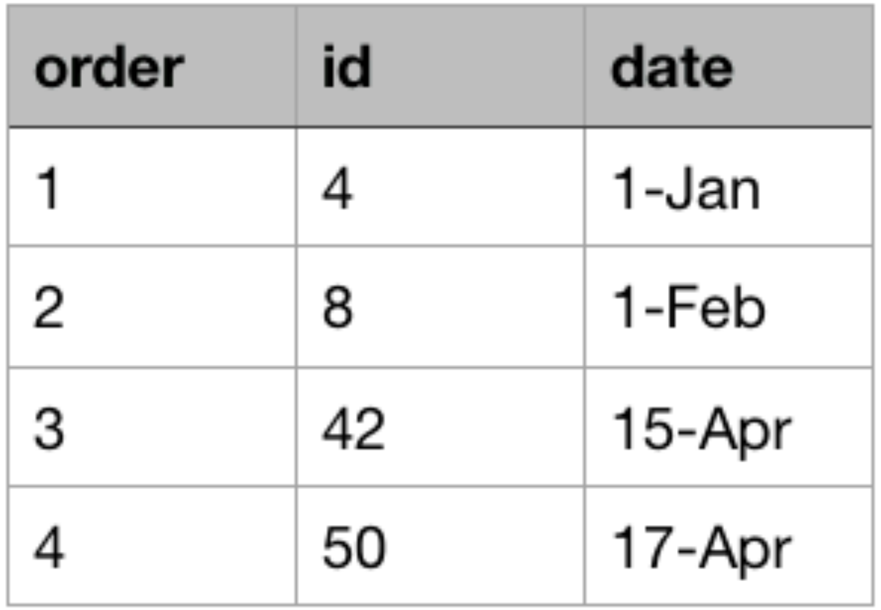
An online retailer stores customer data in two places:

customers



id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders



order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr

Types of joins

1. **Mutating joins** - add new variables to one data frame from matching observations in another

2. **Filtering joins** - filter observations from one data frame based on whether or not they match an observation in the other

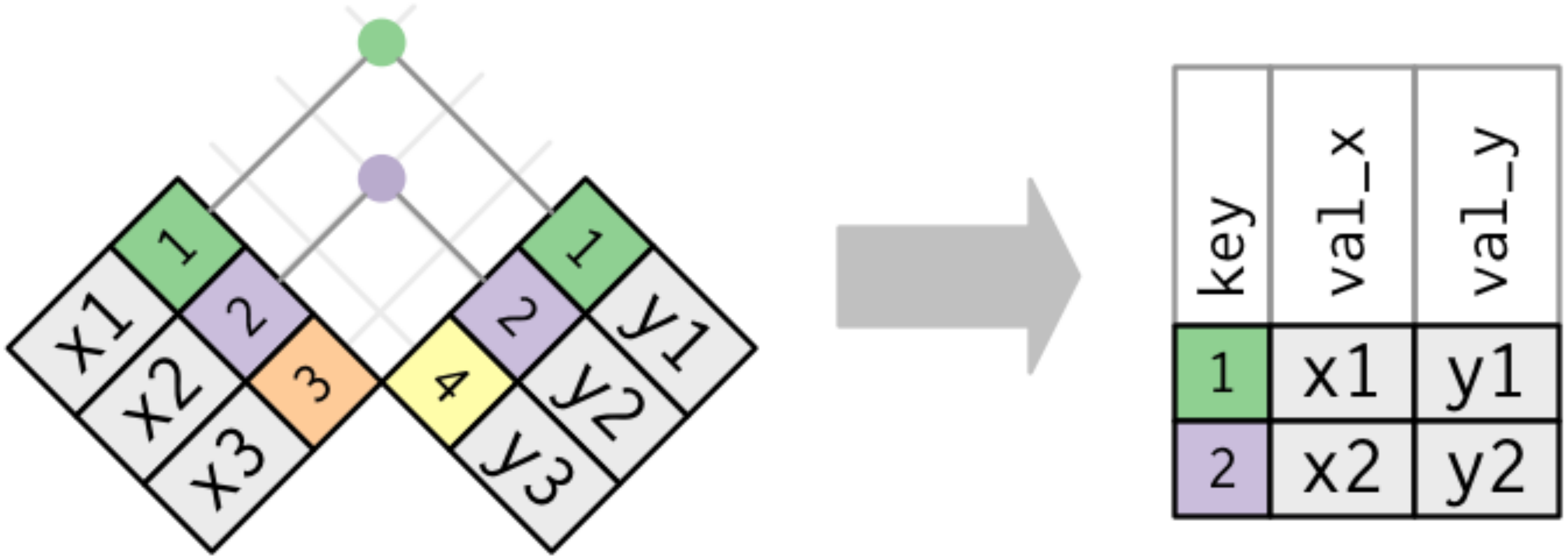
Keys

Used to connect two data tables

- **primary key** uniquely identifies an observation in its own table
- **foreign key** uniquely identifies an observation in another table

`inner_join`

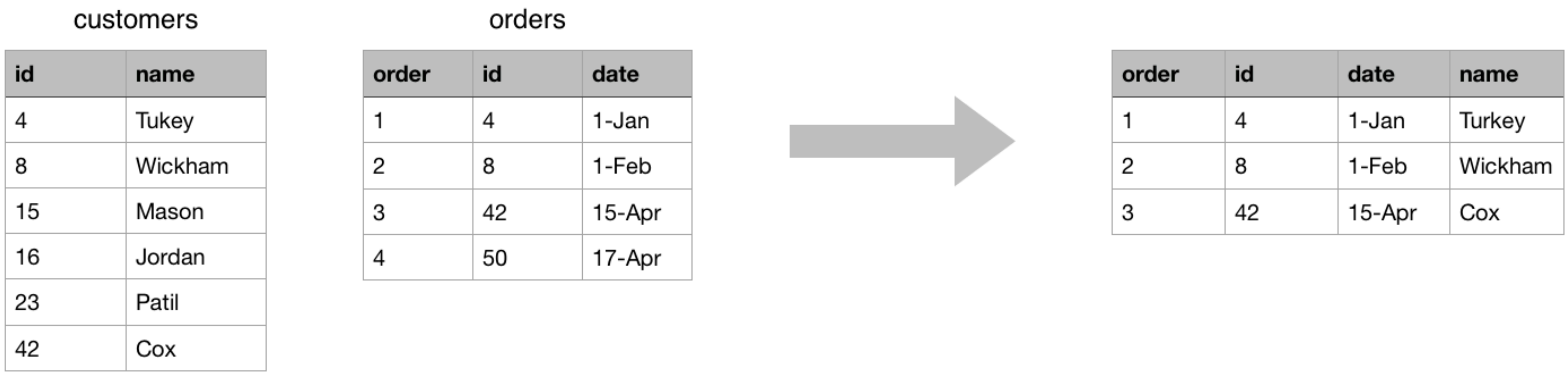
An `inner_join` matches pairs of observations when their keys are equal



key	val_x	val_y
1	x1	y1
2	x2	y2

inner_join

```
inner_join(x = orders, y = customers, by = "id")
```



customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

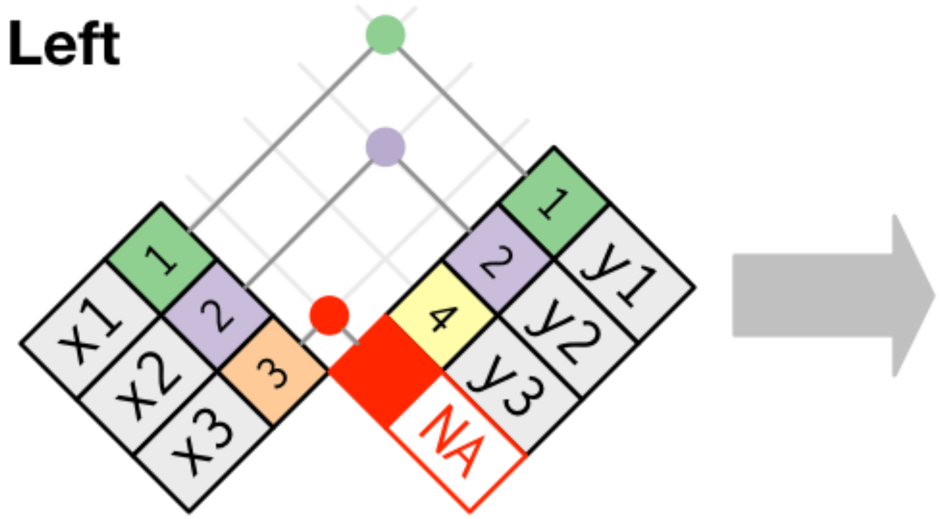
orders

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr

order	id	date	name
1	4	1-Jan	Turkey
2	8	1-Feb	Wickham
3	42	15-Apr	Cox

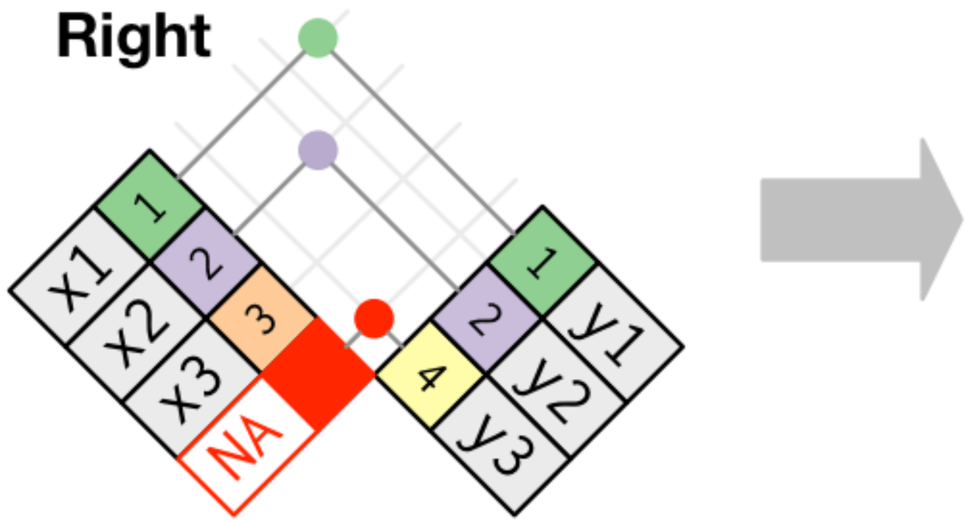
Outer joins

Keep the rows that appear in a specified table



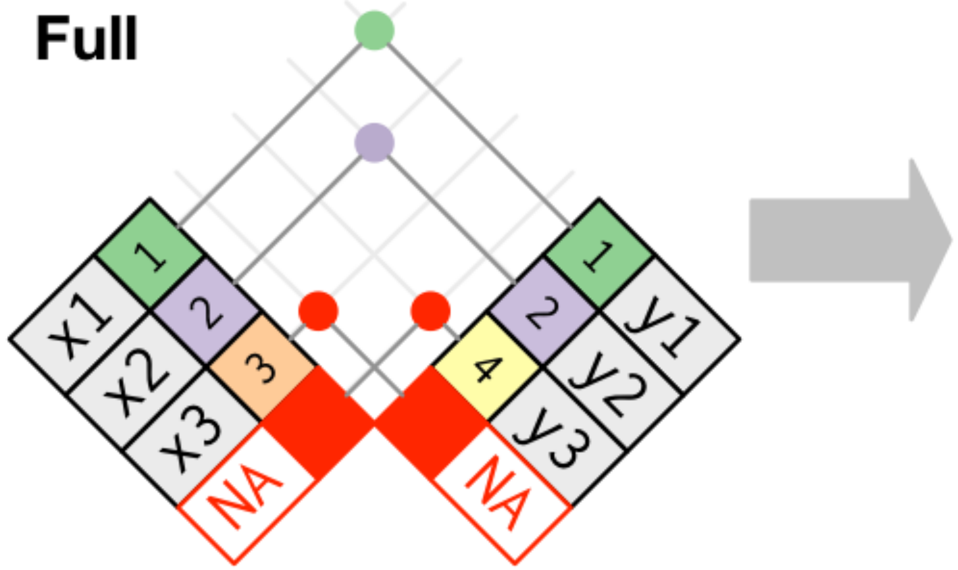
key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

A **left join** keeps all observations in the `x` table



key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

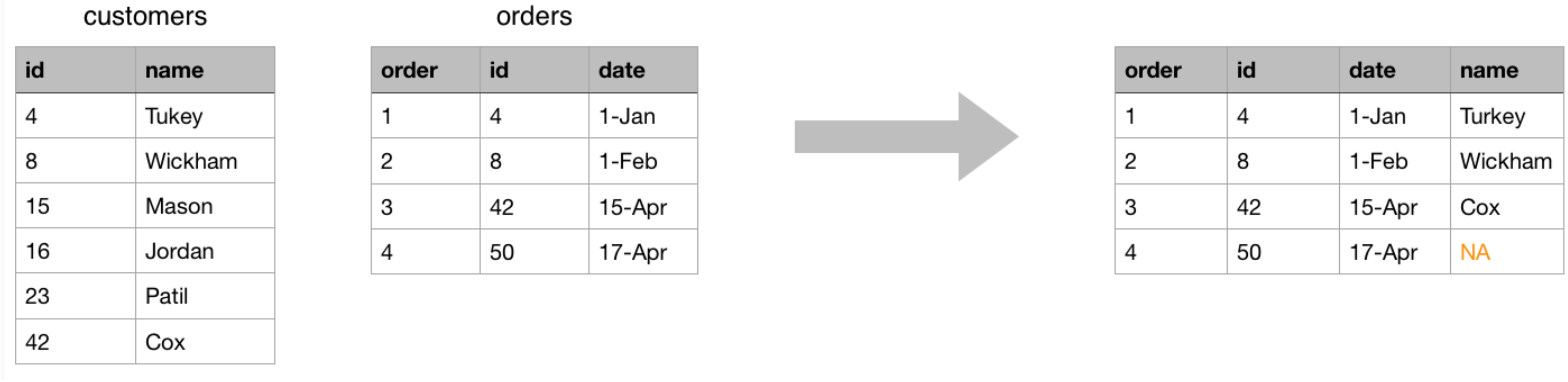
A **right join** keeps all observations in the y table



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA
4	NA	y3

A **full join** keeps all observations in both the `x` and y tables

```
left_join(x = orders, y = customers, by = "id")
```



customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr

order	id	date	name
1	4	1-Jan	Turkey
2	8	1-Feb	Wickham
3	42	15-Apr	Cox
4	50	17-Apr	NA

```
right_join(x = orders, y = customers, by = "id")
```

customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr

→

order	id	date	name
1	4	1-Jan	Turkey
2	8	1-Feb	Wickham
NA	15	NA	Mason
NA	16	NA	Jordan
NA	23	NA	Patil
3	42	15-Apr	Cox

```
full_join(x = orders, y = customers, by = "id")
```

customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr

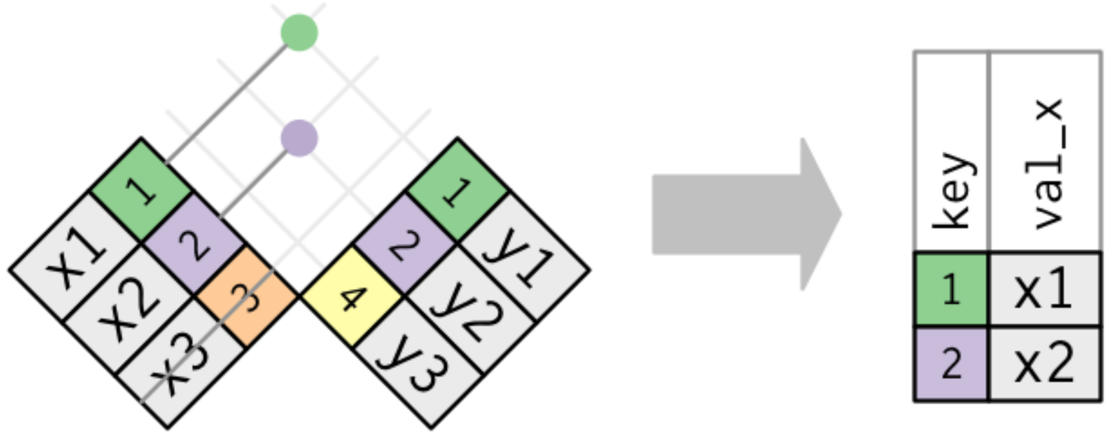
order	id	date	name
1	4	1-Jan	Turkey
2	8	1-Feb	Wickham
3	42	15-Apr	Cox
4	50	17-Apr	NA
NA	15	NA	Mason
NA	16	NA	Jordan
NA	23	NA	Patil
NA	42	NA	Cox

Filtering joins

Filtering joins still match observations between two data tables, but do not add additional variables, they **only impact the rows returned.**

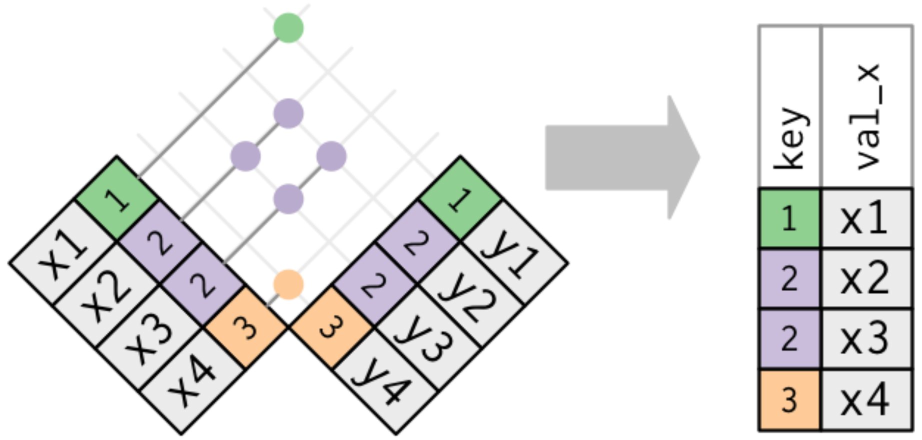
1. `semi_join(x, y)` keeps all observations in `x` that have a match in y
2. `anti_join(x, y)` drops all observations in `x` that have a match in y

semi_join(x, y)



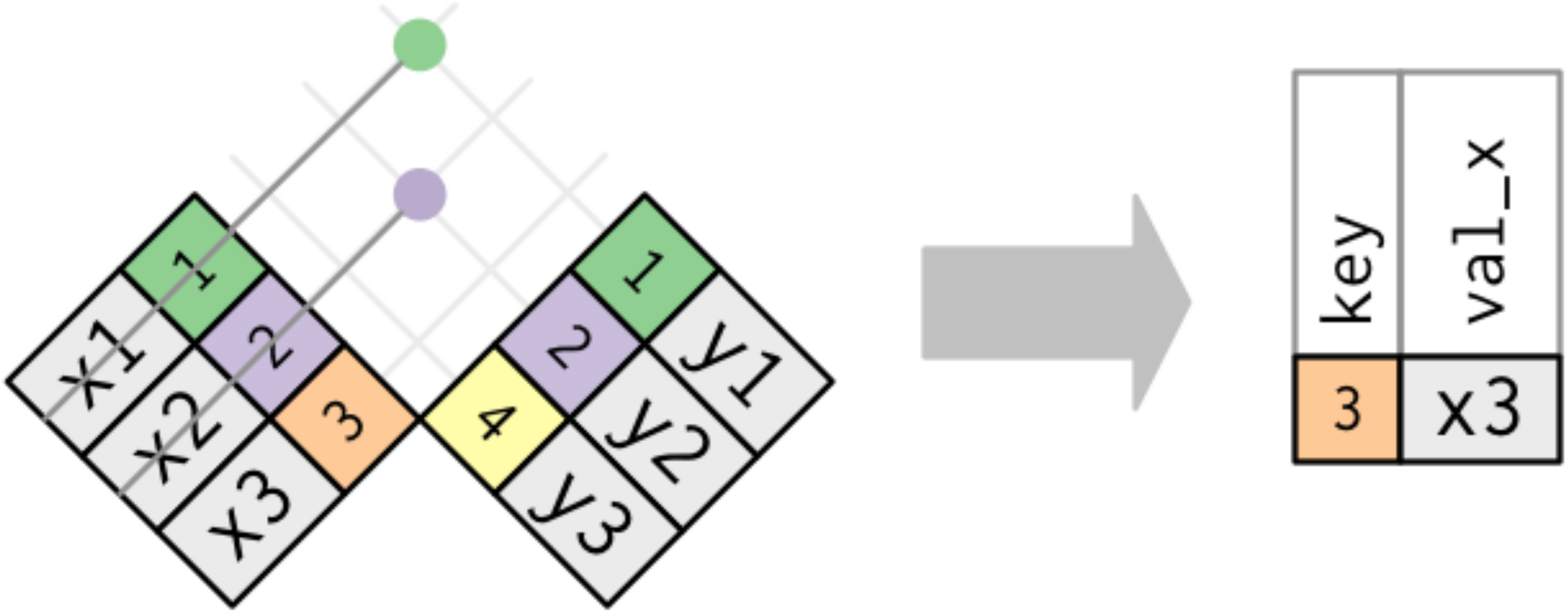
key	val_x
1	x1
2	x2

Observations will never be duplicated.



key	val_x
1	x1
2	x2
2	x3
3	x4

anti_join(x, y)



key	val_x
3	x3

```
semi_join(x = orders, y = customers, by = "id")
```

orders

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr

customers

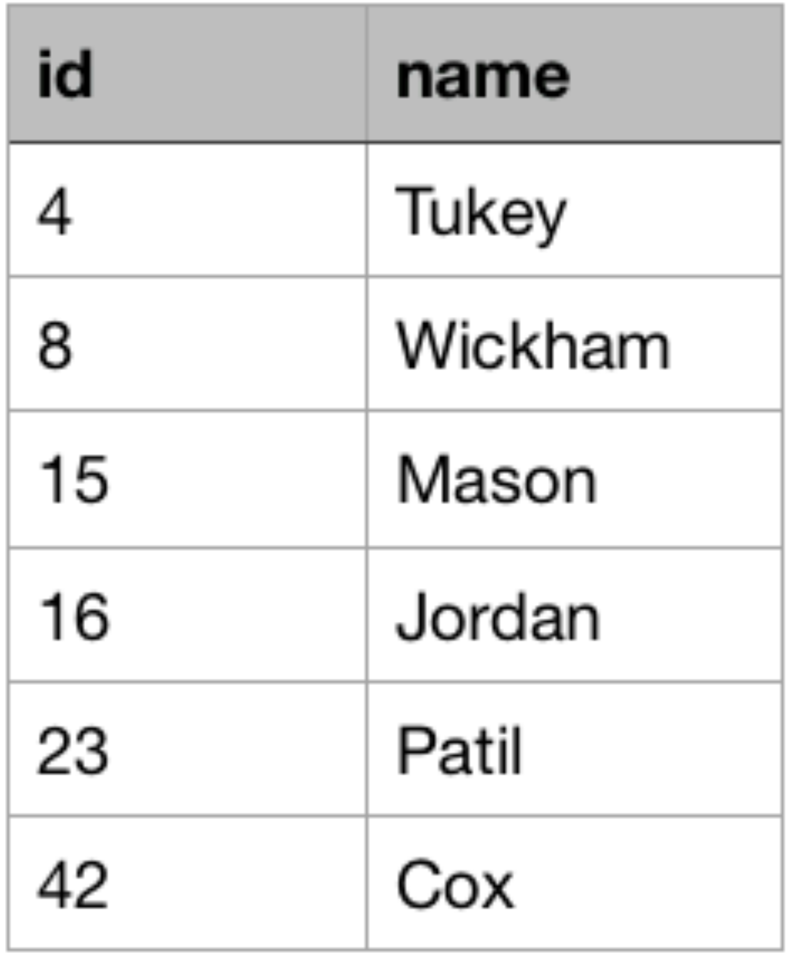
id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr

What if we had an extra order?

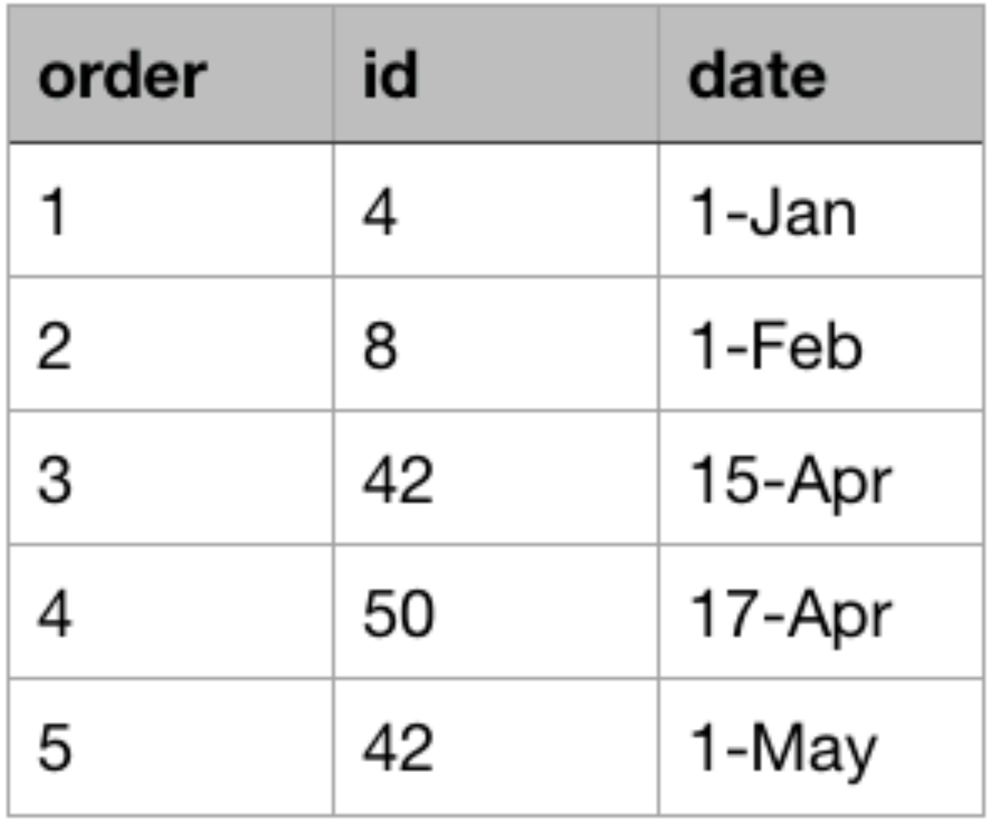
```
extra_order <- data.frame(order = 5, id = 42, date = "May-01")
orders2 <- rbind(orders, extra_order)
```

customers



id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

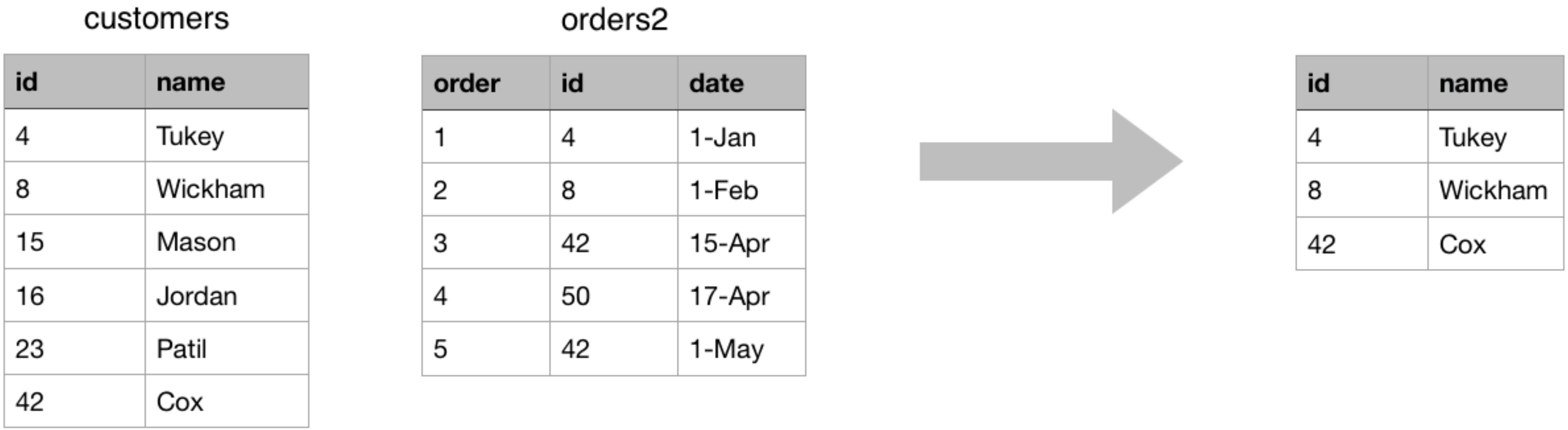
orders2



order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr
5	42	1-May

How do `semi_join` and `inner_join` differ?

```
semi_join(x = customers, y = orders2, by = "id")
```



customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders2

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr
5	42	1-May

id	name
4	Tukey
8	Wickham
42	Cox

How do semi_join and inner_join differ?

```
inner_join(x = customers, y = orders2, by = "id")
```

customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

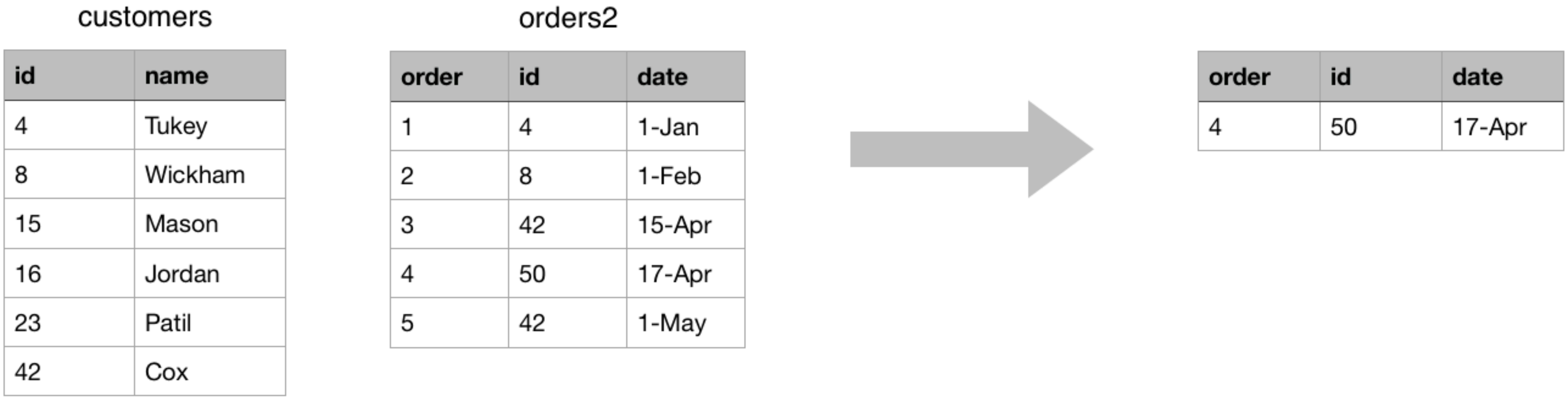
orders2

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr
5	42	1-May

id	date	name
4	1-Jan	Turkey
8	1-Feb	Wickham
42	15-Apr	Cox
42	1-May	Cox

For an anti_join, order matters

```
anti_join(x = orders2, y = customers, by = "id")
```



customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders2

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr
5	42	1-May

order	id	date
4	50	17-Apr

For an `anti_join`, order matters

```
anti_join(x = customers, y = orders2, by = "id")
```

customers

id	name
4	Tukey
8	Wickham
15	Mason
16	Jordan
23	Patil
42	Cox

orders2

order	id	date
1	4	1-Jan
2	8	1-Feb
3	42	15-Apr
4	50	17-Apr
5	42	1-May

id	name
23	Patil
16	Jordan
15	Cox

Common complications

Joining by multiple variables,

You must specify a vector of variable names:

```
by = c("var1", "var2", "var3").
```

All three columns must match in both tables.

Use all variables that appear in both tables

Leave the `by` argument blank.

Column names differ between tables

```
by = c("left_var" = "right_var").
```

Your Turn

Does payroll differ between the American League and the National League?

- Load the `tidyverse`
- Install and load the `Lahman` R package
- Look at the `Salaries` and `Teams` data tables
- Devise a way to clearly compare the team payroll between the two leagues over the years