PSY 503: Foundations of Statistical Methods in Psychological Science

LAB: Joins, Broom,

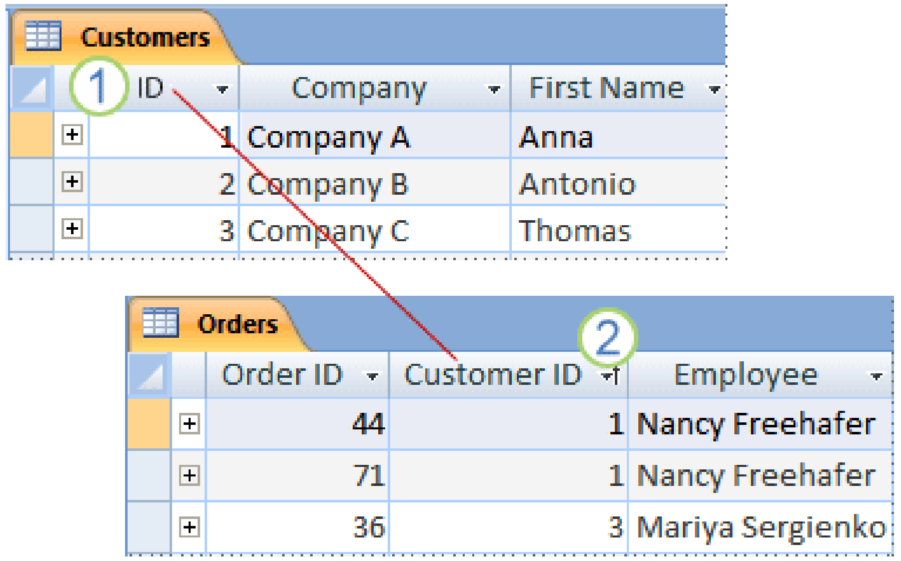
Regression Intro

Suyog Chandramouli

Zoom & 311 PSH (Princeton University)

1st October, 2025

ID / Primary key



Customers

	ID	Company	First Name
	1	Company A	Anna
	2	Company B	Antonio
	3	Company C	Thomas

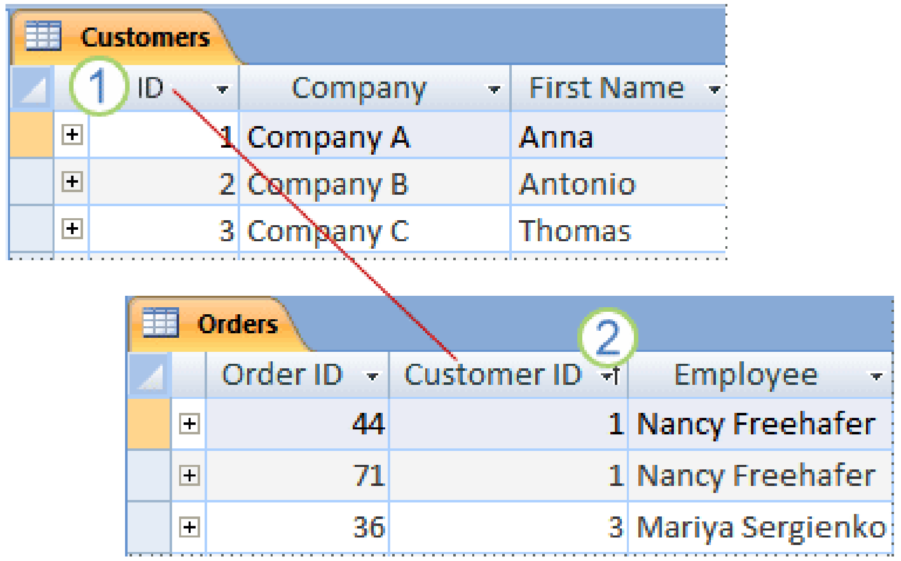
Orders

	Order ID	Customer ID	Employee
	44	1	Nancy Freehafer
	71	1	Nancy Freehafer
	36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.



Customers

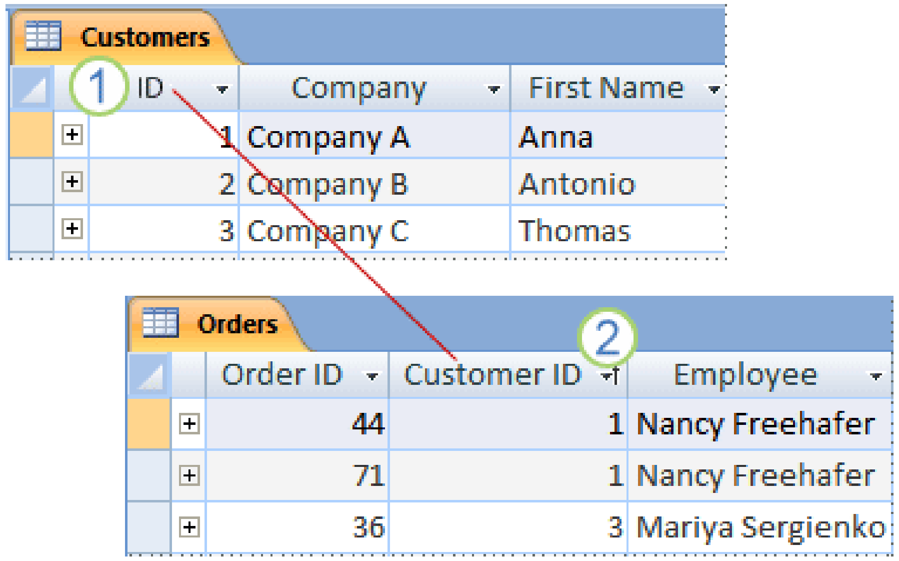
ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

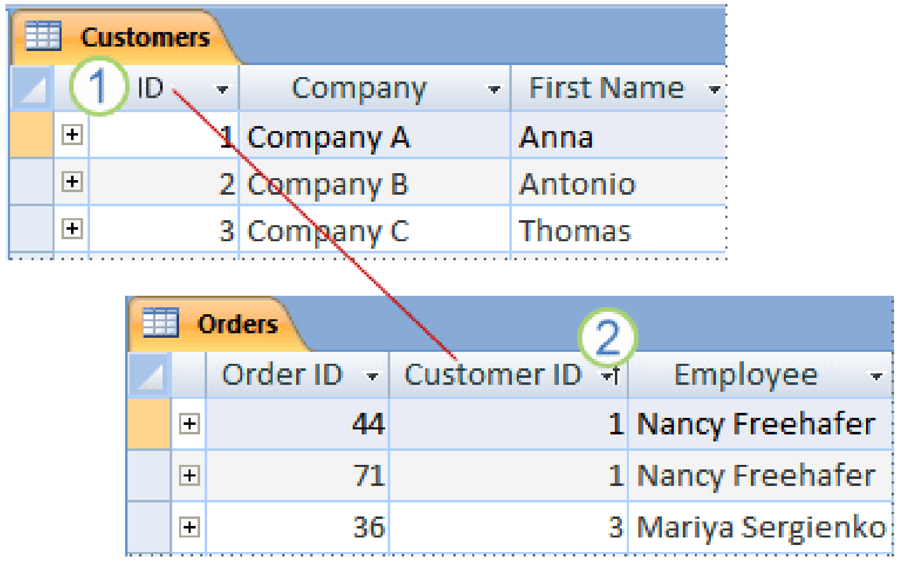
Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.

- Seldom are all necessary data in a single dataset



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

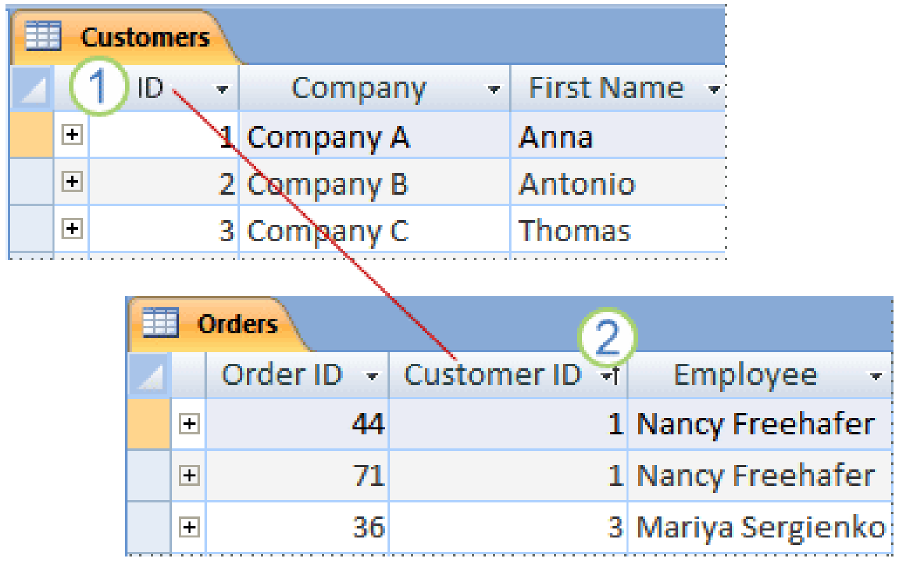
Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.

- Seldom are all necessary data in a single dataset
  - E.g.
    - Govt Source A: COVID-19 daily case counts by region
    - Govt Source B: Vaccination rates by region



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

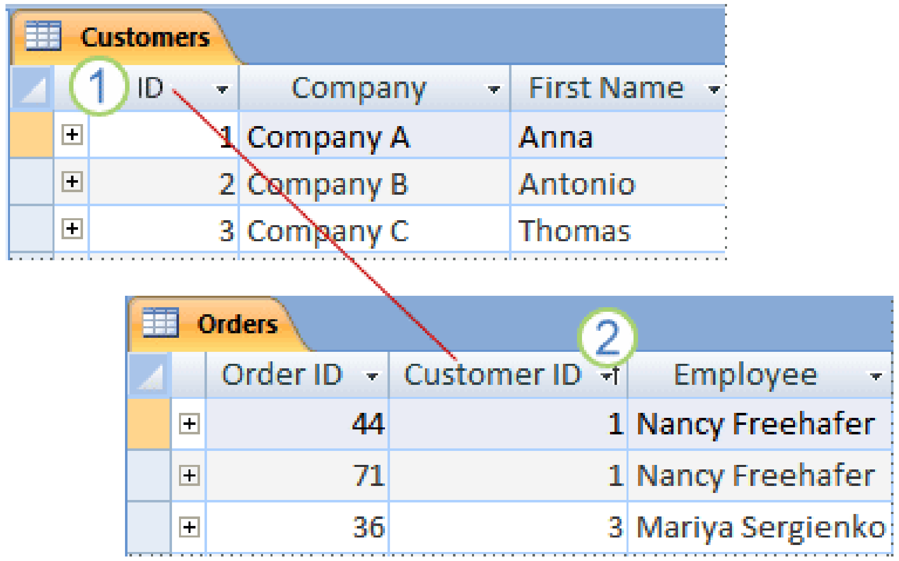
Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.

- Seldom are all necessary data in a single dataset
  - E.g.
    - Govt Source A: COVID-19 daily case counts by region
    - Govt Source B: Vaccination rates by region
  - **Join:** For analyzing the correlation between vaccination rates and case numbers



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

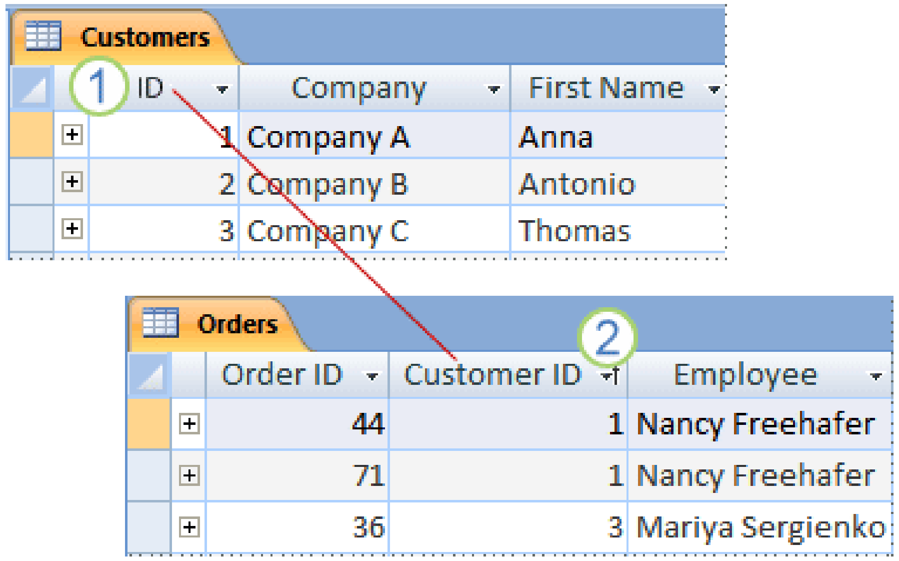
Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.

- Seldom are all necessary data in a single dataset
  - E.g.
    - Govt Source A: COVID-19 daily case counts by region
    - Govt Source B: Vaccination rates by region
  - **Join:** For analyzing the correlation between vaccination rates and case numbers



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

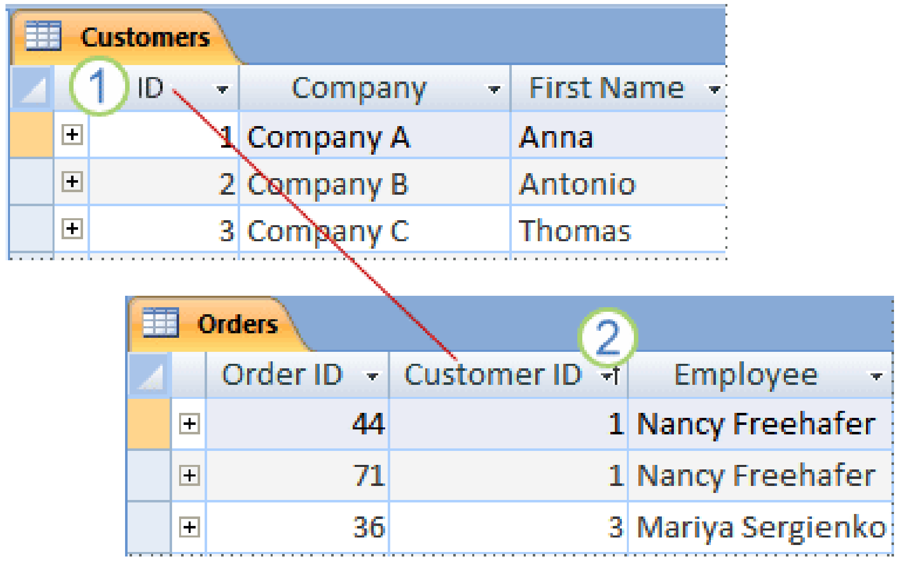
Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.

- Seldom are all necessary data in a single dataset



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

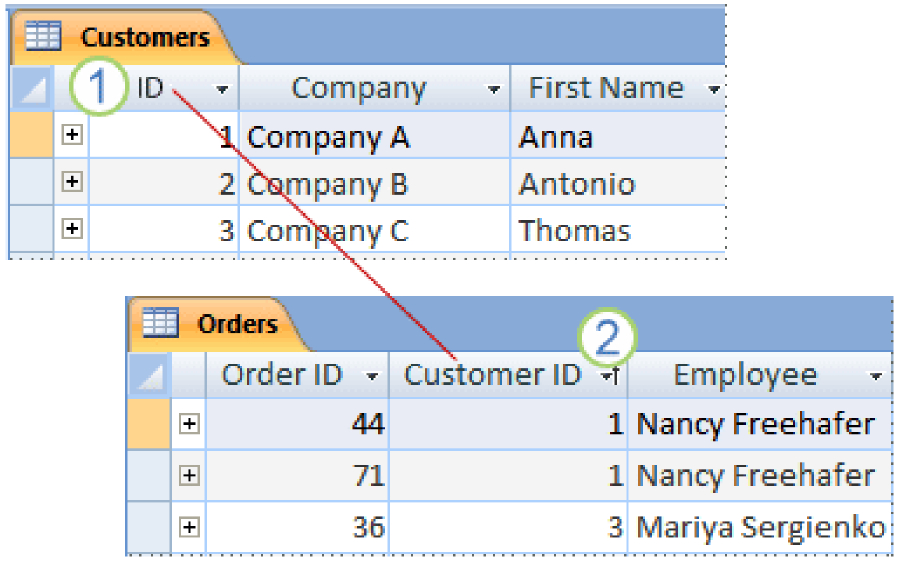
Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

ID / Primary key

- a column that is distinct for each record
  - SSN
  - Princeton NetID
  - Username

  , etc.

- Seldom are all necessary data in a single dataset
  - E.g. Psychology experiment
    - A: Demographic information
    - B: Cognitive Tests data
  - **Join:** To get the complete picture and to look at association with demographic information



Customers

ID	Company	First Name
1	Company A	Anna
2	Company B	Antonio
3	Company C	Thomas

Orders

Order ID	Customer ID	Employee
44	1	Nancy Freehafer
71	1	Nancy Freehafer
36	3	Mariya Sergienko

Joins

- Different ways to merge dataframes
- Around one or more identifying variables

- Interface:
    - Input: Two dataframes
    - Output: One dataframe

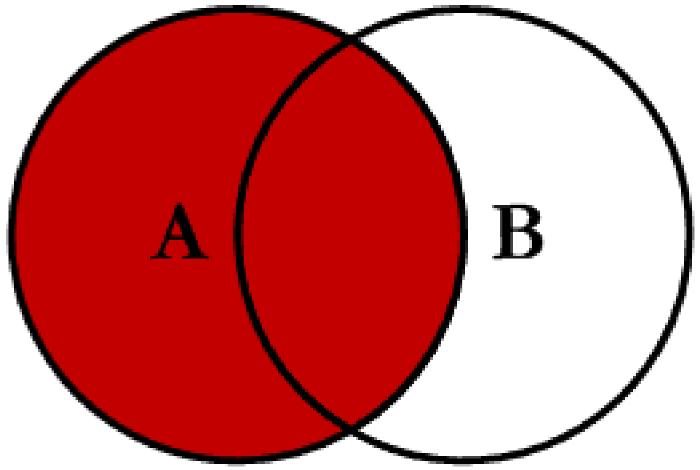
Joins

- Different ways to merge dataframes
- Around one or more identifying variables

- Interface:
  - Input: Two dataframes
  - Output: One dataframe
    (different join types keep different rows)

Left Join

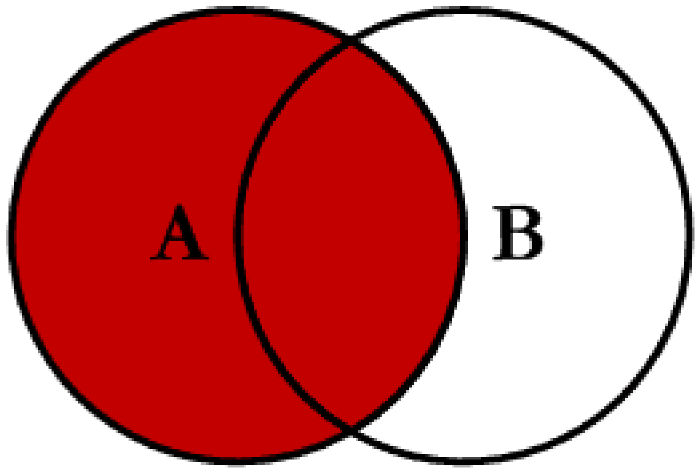
- Keep all of Left Data



Source: https://ugoproto.github.io/ugodoc/introduction_to_sql_joins/

Left Join

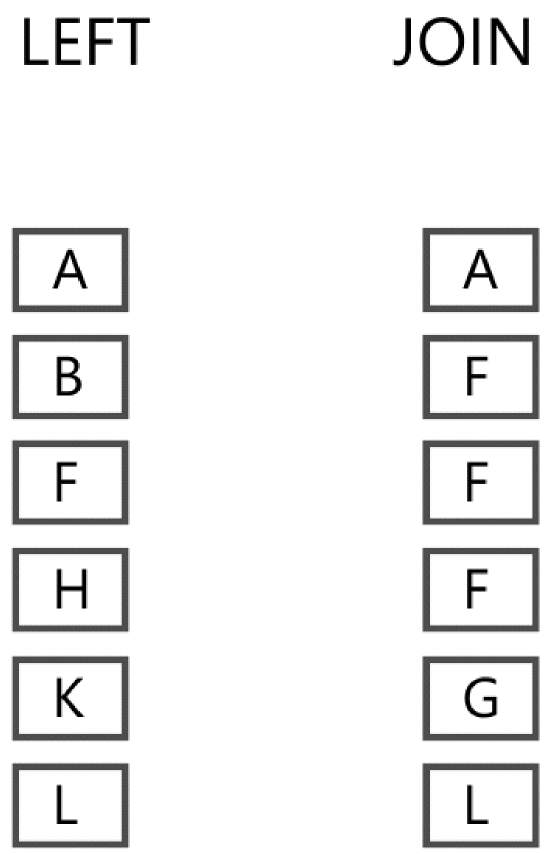
- Keep all of Left Data
- Result has same number of rows as Data on the Left



Source: https://ugoproto.github.io/ugodoc/introduction_to_sql_joins/

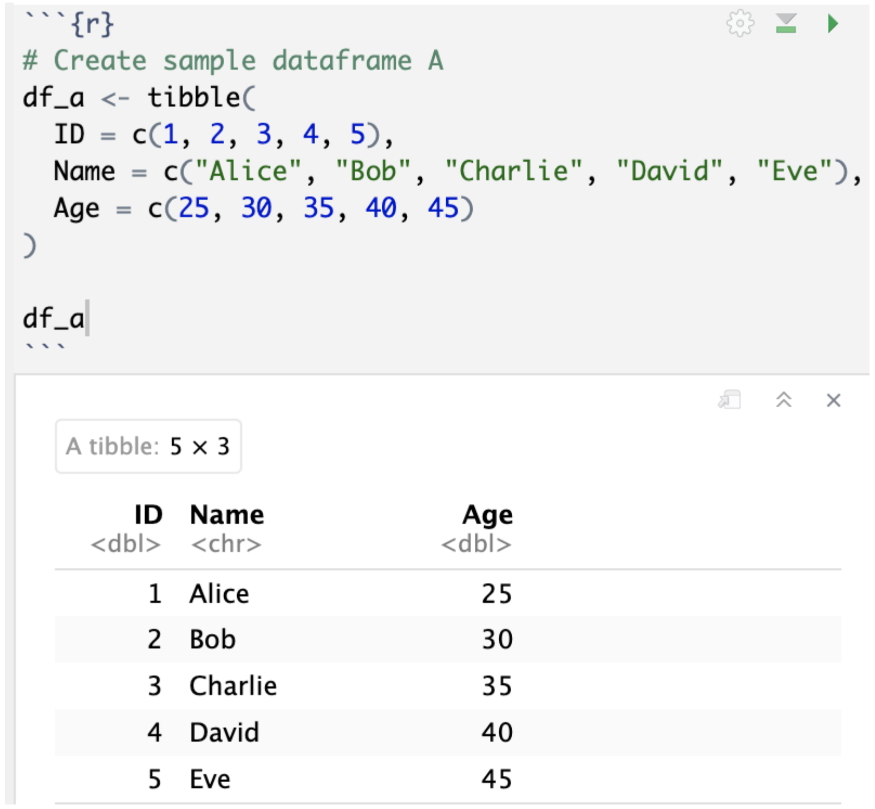
Left Join

- Keep all of Left Data
- Result has same number of rows as Data on the Left



Source: https://www.reddit.com/r/SQL/comments/17ohvjc/join_visualizations_that_arent_venn_diagrams/

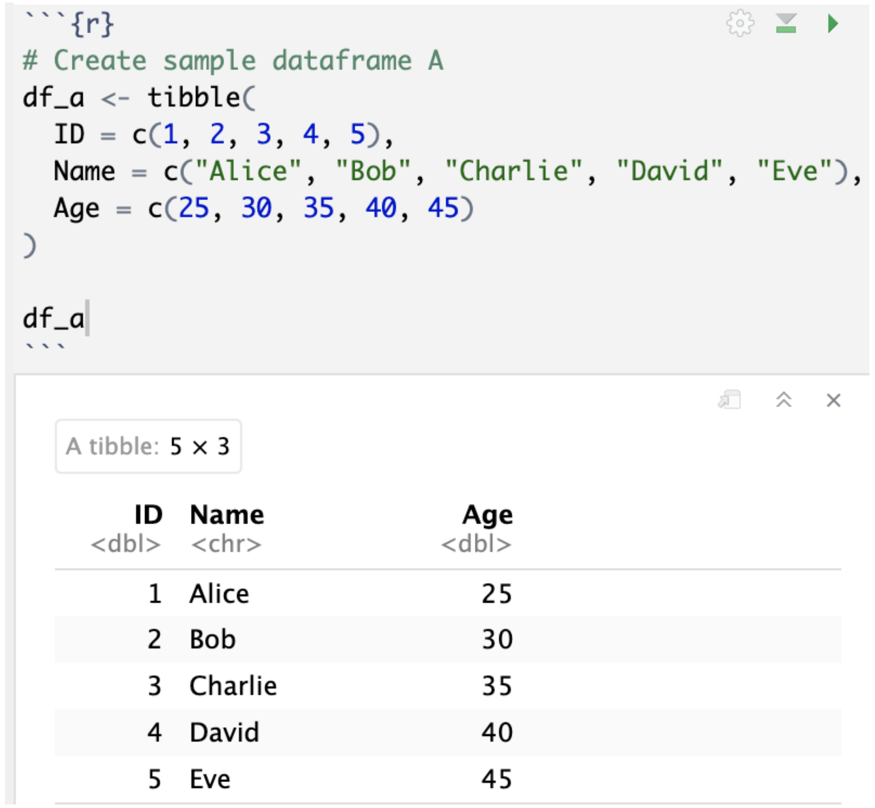
Left Join (Exercise)



~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

```
A tibble: 5 × 3

   ID Name        Age
<dbl> <chr>     <dbl>
    1 Alice        25
    2 Bob          30
    3 Charlie      35
    4 David        40
    5 Eve          45
```

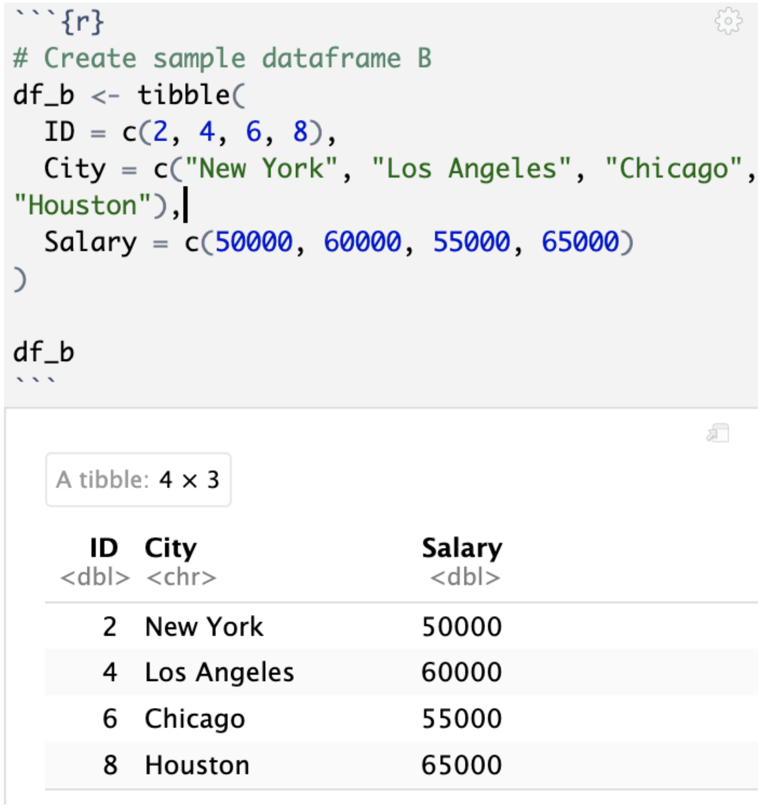
Left Join (Exercise)



~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

A tibble: 5 × 3

ID <dbl>	Name <chr>	Age <dbl>
1	Alice	25
2	Bob	30
3	Charlie	35
4	David	40
5	Eve	45

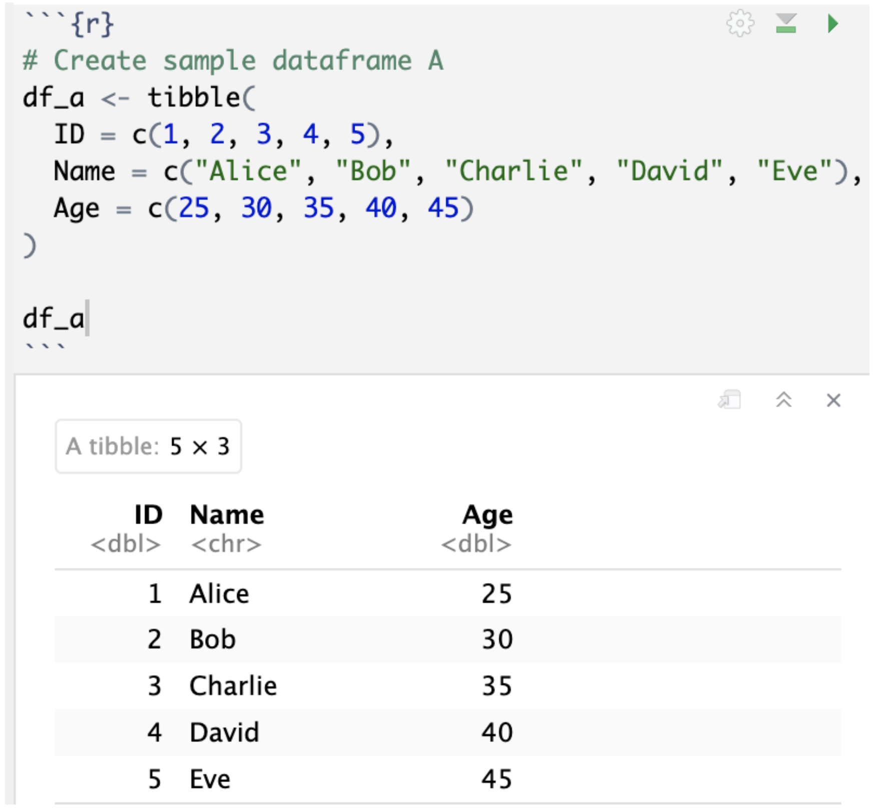


~~~
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```
~~~

A tibble: 4 × 3

ID <dbl>	City <chr>	Salary <dbl>
2	New York	50000
4	Los Angeles	60000
6	Chicago	55000
8	Houston	65000

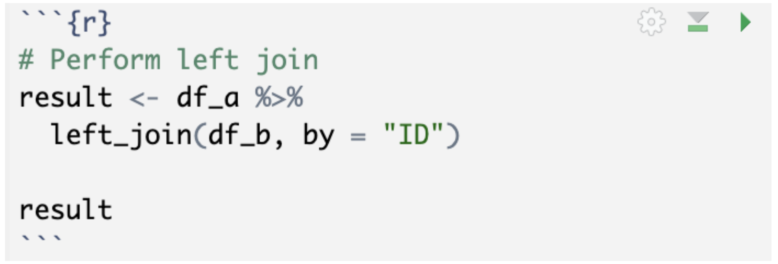
Left Join (Exercise)



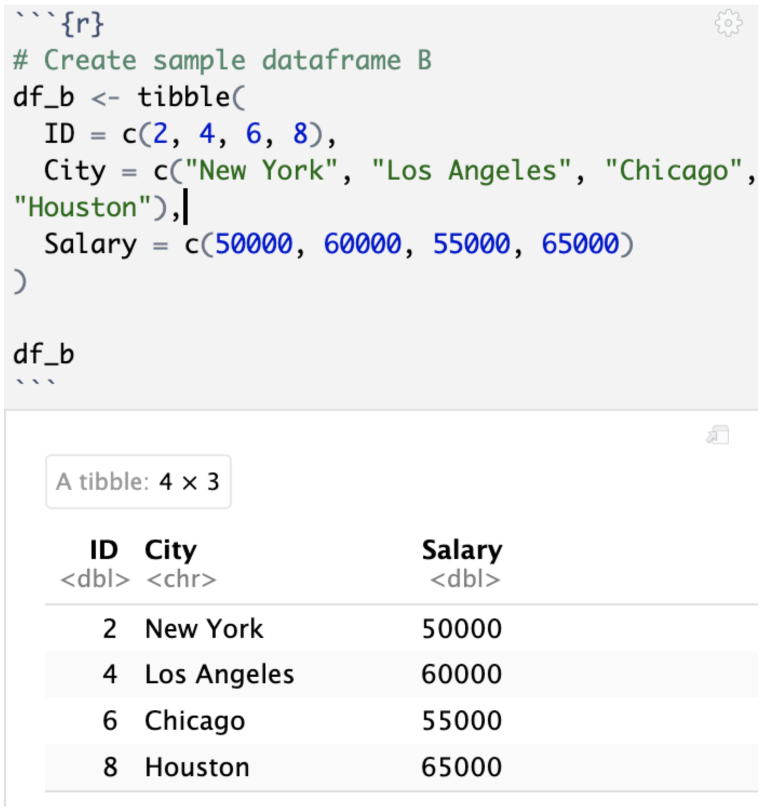
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```

A tibble: 5 × 3

ID <dbl>	Name <chr>	Age <dbl>
1	Alice	25
2	Bob	30
3	Charlie	35
4	David	40
5	Eve	45



```{r}
# Perform left join
result <- df_a %>%
  left_join(df_b, by = "ID")

result
```

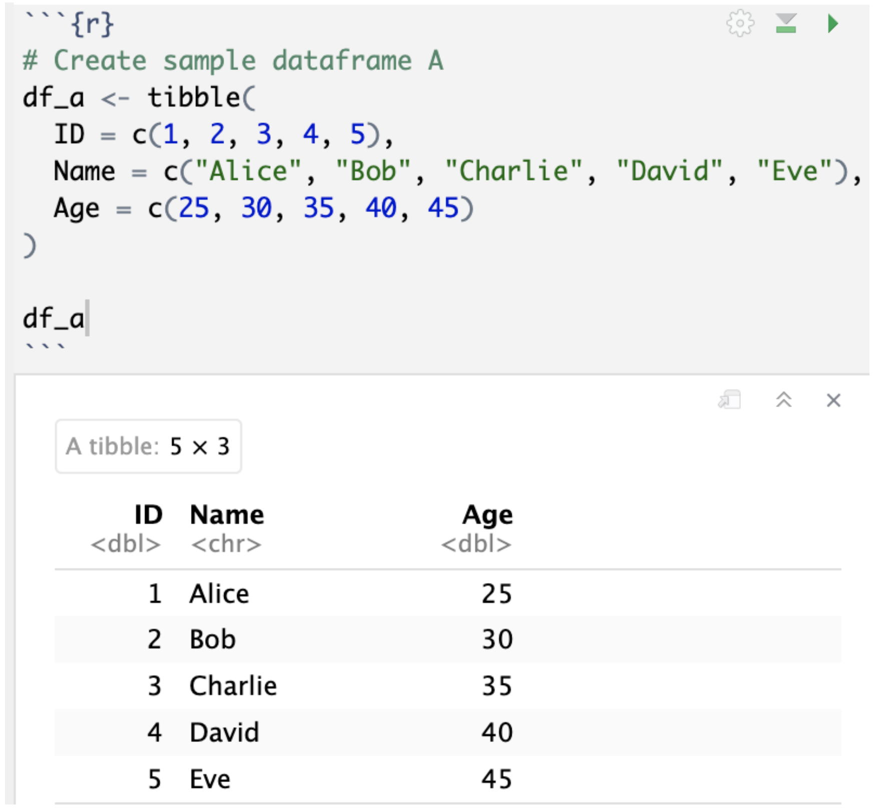


```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```

A tibble: 4 × 3

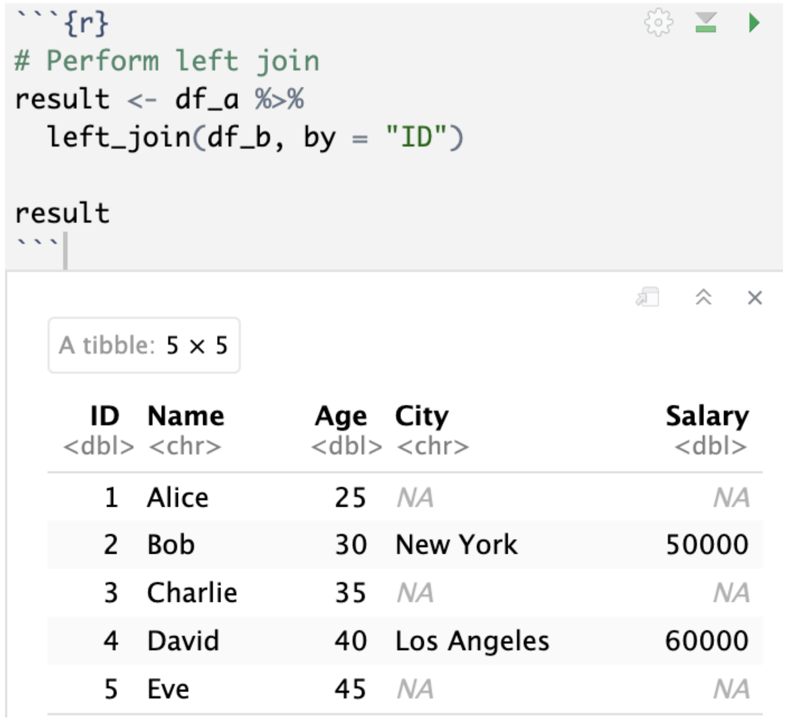
ID <dbl>	City <chr>	Salary <dbl>
2	New York	50000
4	Los Angeles	60000
6	Chicago	55000
8	Houston	65000

Left Join (Exercise)



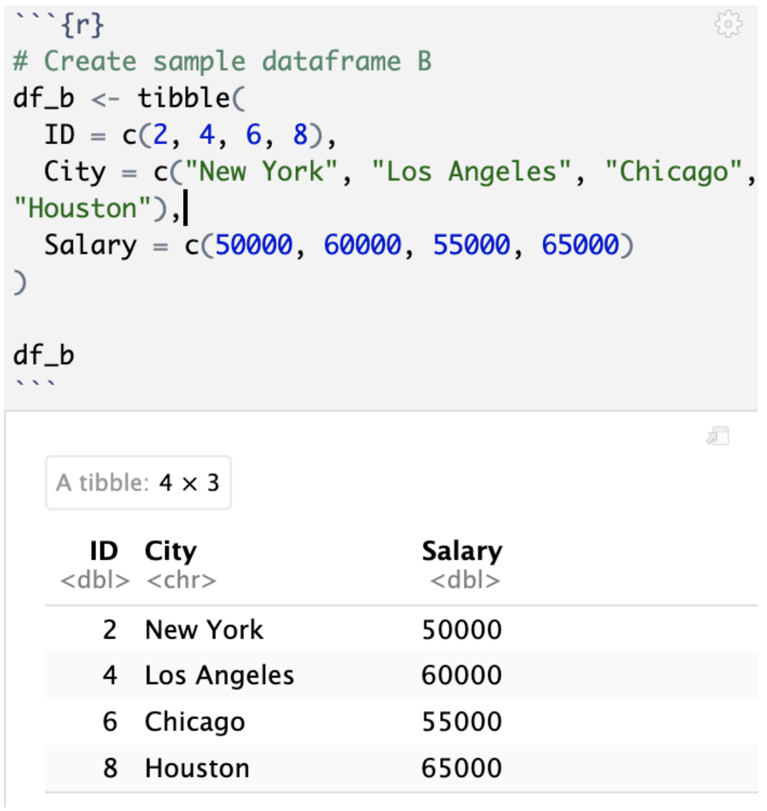
~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

```
A tibble: 5 × 3

   ID  Name       Age
<dbl>  <chr>    <dbl>
    1  Alice       25
    2  Bob         30
    3  Charlie     35
    4  David       40
    5  Eve         45
```



~~~
```{r}
# Perform left join
result <- df_a %>%
  left_join(df_b, by = "ID")

result
```
~~~

```
A tibble: 5 × 5

   ID  Name       Age  City          Salary
<dbl>  <chr>    <dbl>  <chr>          <dbl>
    1  Alice       25  NA                NA
    2  Bob         30  New York       50000
    3  Charlie     35  NA                NA
    4  David       40  Los Angeles    60000
    5  Eve         45  NA                NA
```

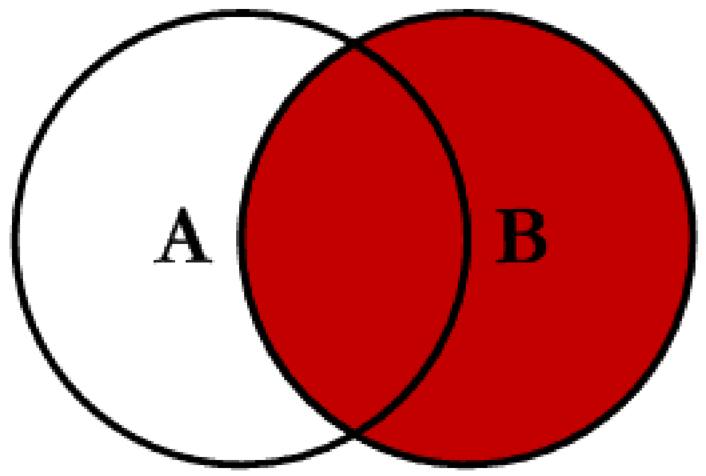


~~~
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```
~~~

```
A tibble: 4 × 3

   ID  City          Salary
<dbl>  <chr>          <dbl>
    2  New York       50000
    4  Los Angeles    60000
    6  Chicago        55000
    8  Houston        65000
```

Right Join

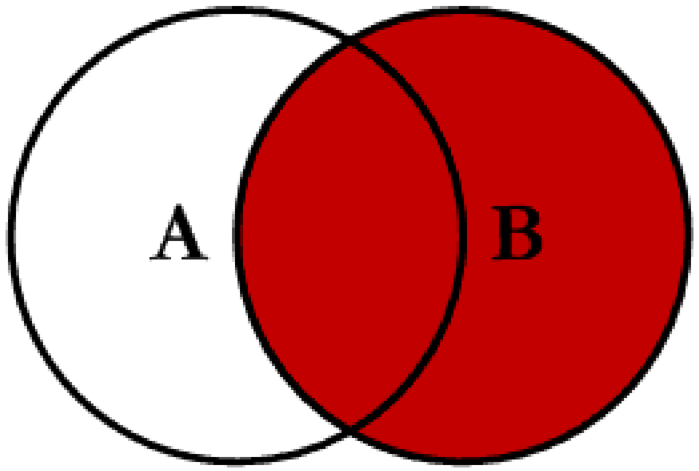
- Keeps all of Right Data



Right Join

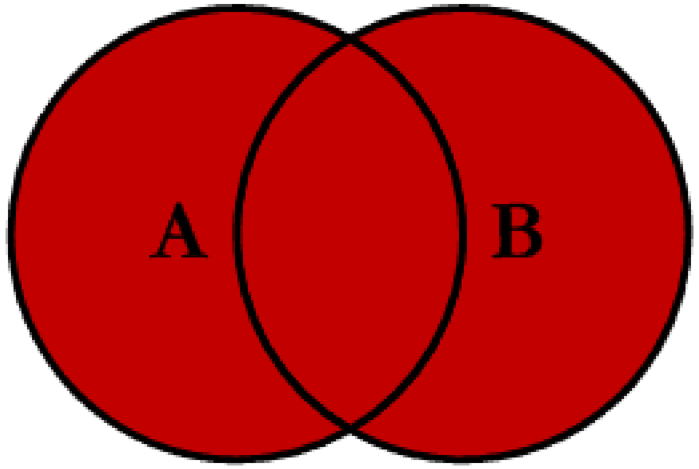
- Keeps all of Right Data
- Same as left join with the sides of the data swapped

- Rarely used. Left join is the norm. (Best to ignore this command)

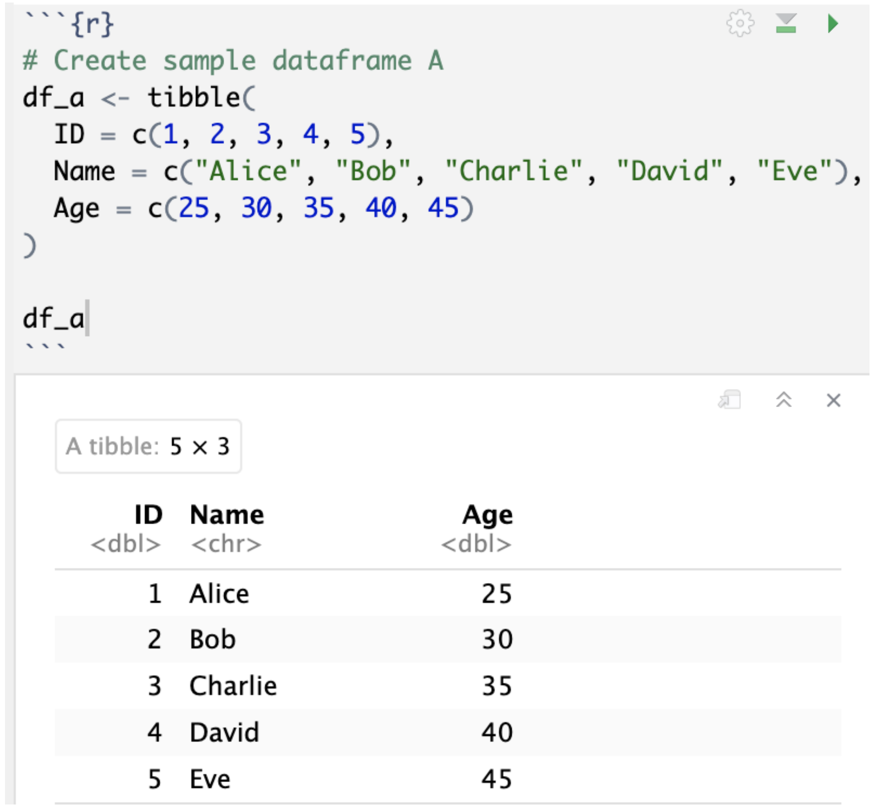


Full Join

- Keep all of Left and Right data
  - Even the ones where there is no overlap



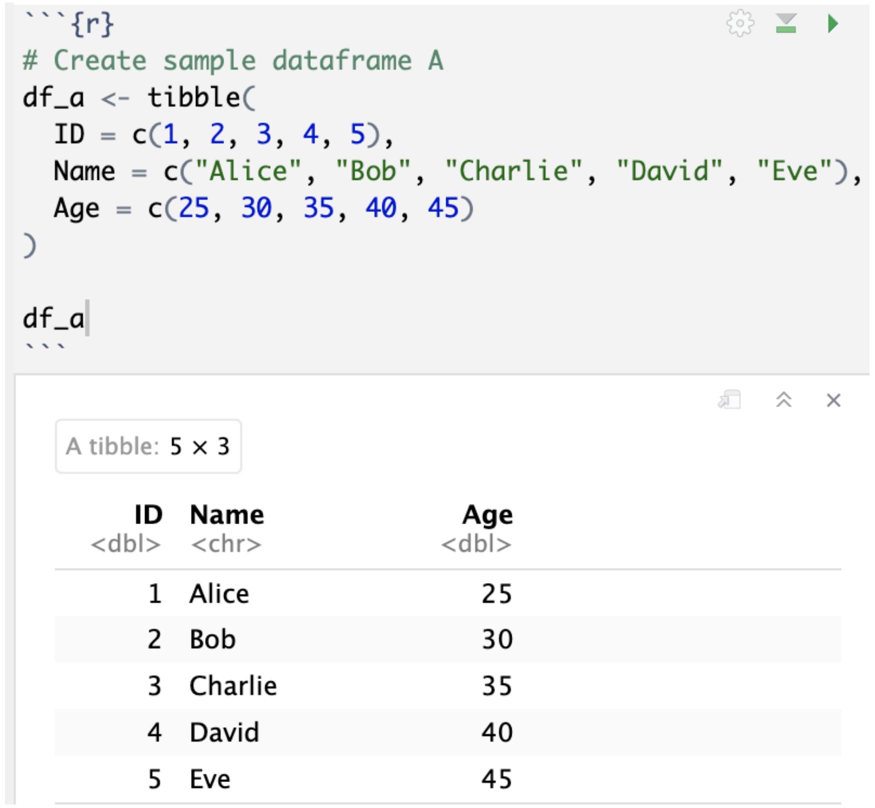
Full Join (Exercise)



~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

```
A tibble: 5 × 3

   ID  Name        Age
<dbl>  <chr>     <dbl>

    1  Alice        25
    2  Bob          30
    3  Charlie      35
    4  David        40
    5  Eve          45
```

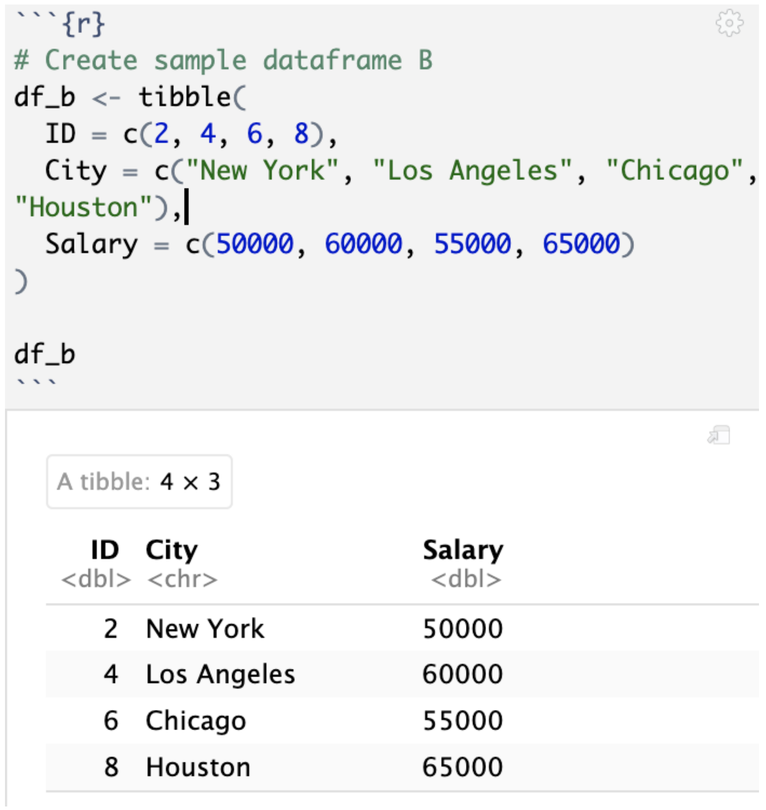
Full Join (Exercise)



```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```

A tibble: 5 × 3

ID <dbl>	Name <chr>	Age <dbl>
1	Alice	25
2	Bob	30
3	Charlie	35
4	David	40
5	Eve	45

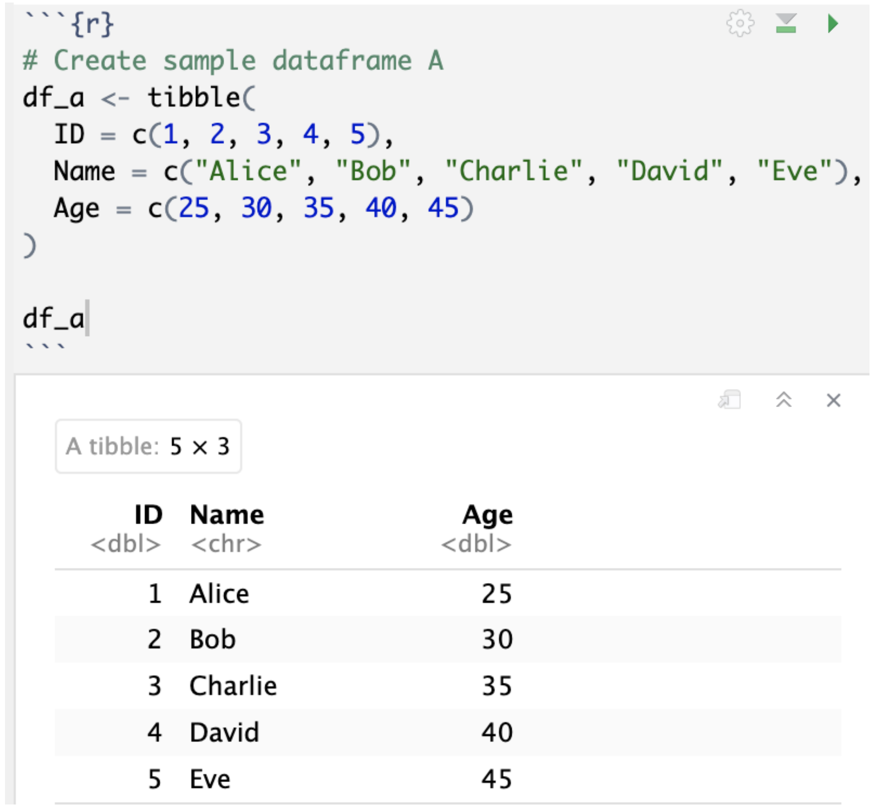


```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```

A tibble: 4 × 3

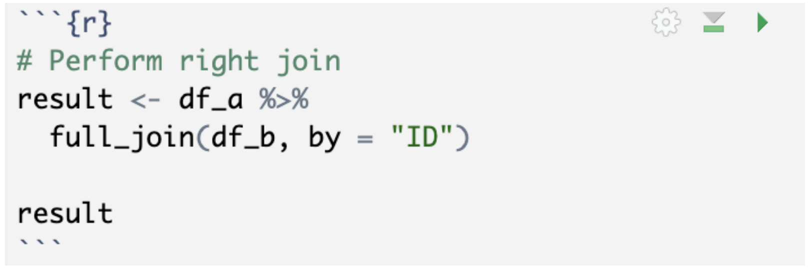
ID <dbl>	City <chr>	Salary <dbl>
2	New York	50000
4	Los Angeles	60000
6	Chicago	55000
8	Houston	65000

Full Join (Exercise)

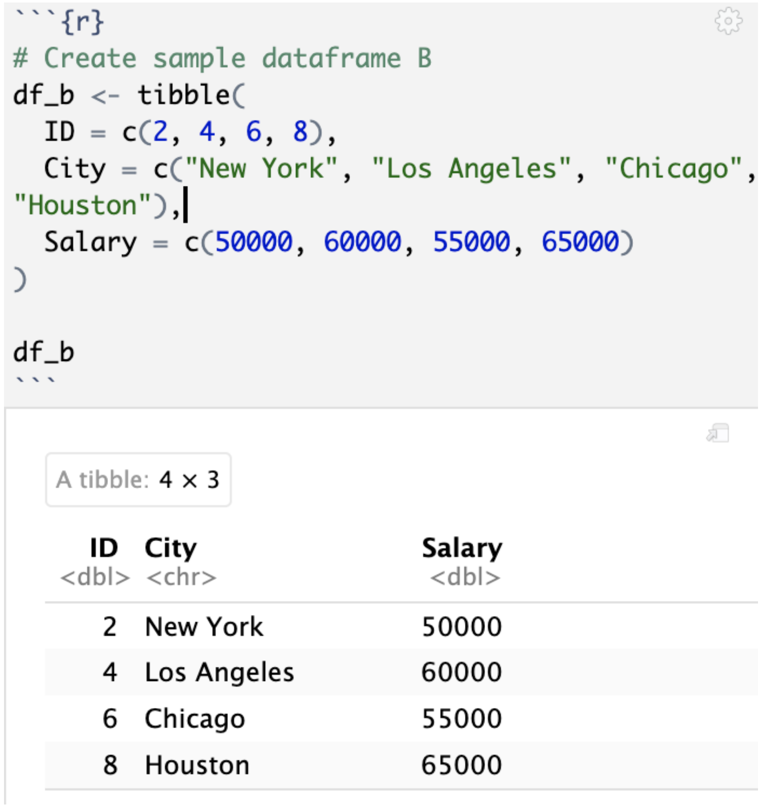


~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

```
A tibble: 5 × 3

   ID Name      Age
<dbl> <chr>   <dbl>
    1 Alice      25
    2 Bob        30
    3 Charlie    35
    4 David      40
    5 Eve        45
```



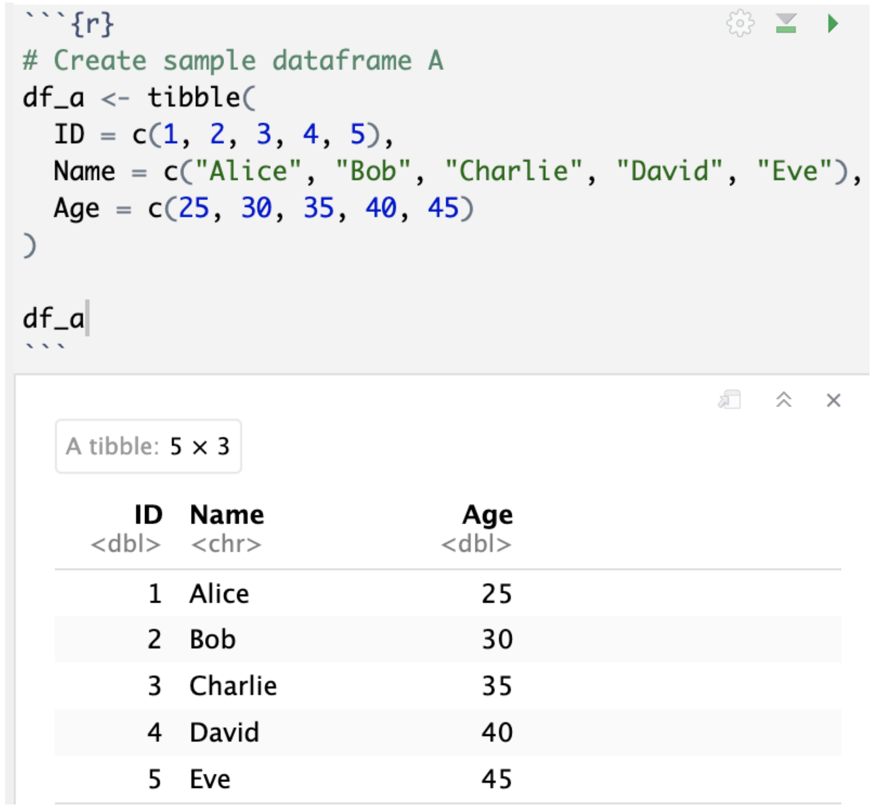
~~~
```{r}
# Perform right join
result <- df_a %>%
  full_join(df_b, by = "ID")

result
```
~~~



~~~
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```
~~~

```
A tibble: 4 × 3

   ID City         Salary
<dbl> <chr>         <dbl>
    2 New York      50000
    4 Los Angeles   60000
    6 Chicago       55000
    8 Houston       65000
```

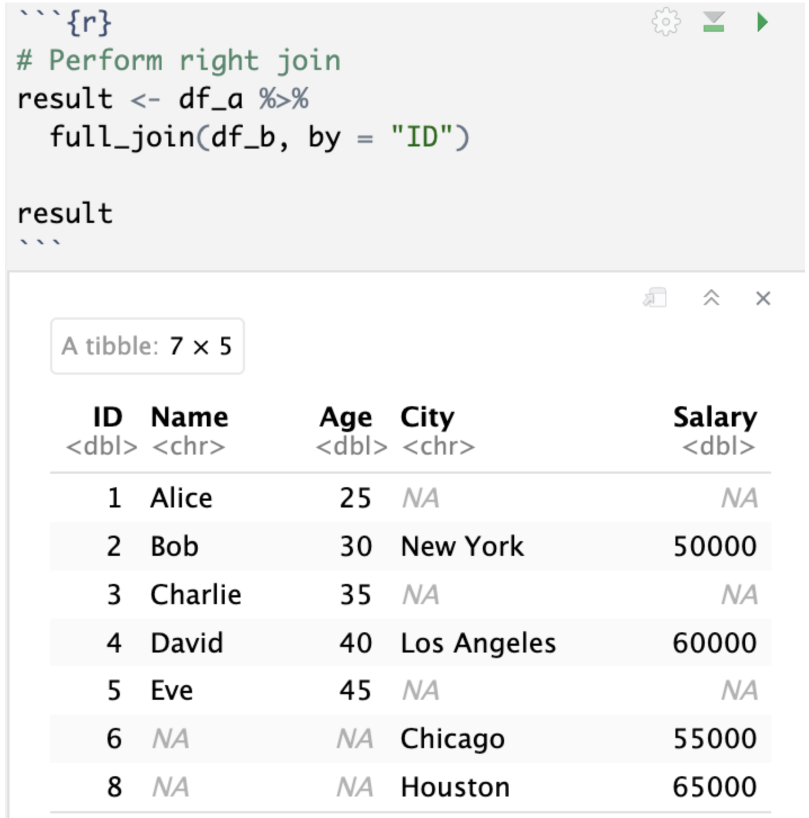
Full Join (Exercise)



```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```

A tibble: 5 × 3

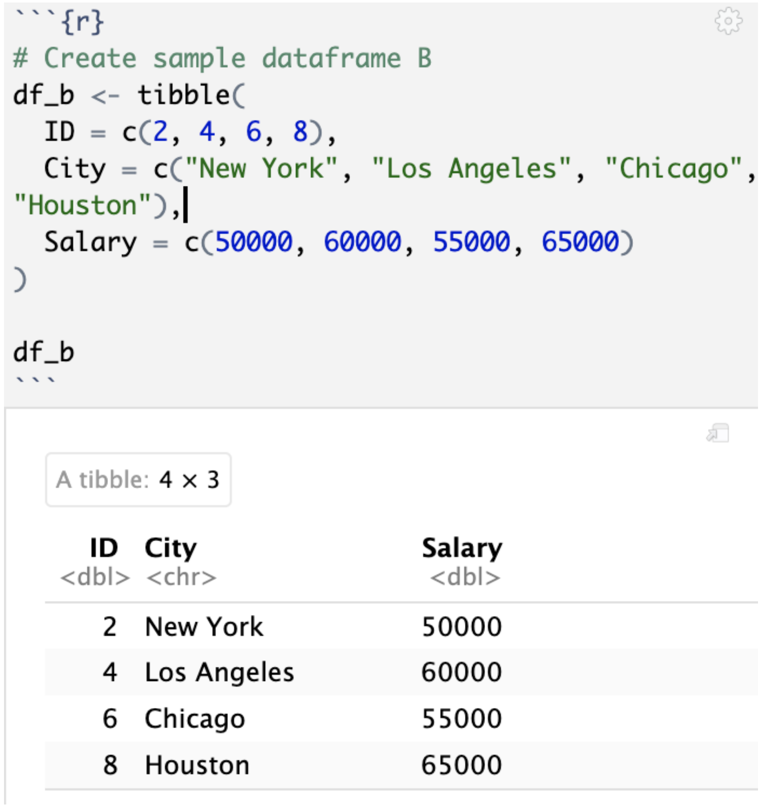
```
 ID   Name      Age
<dbl> <chr>    <dbl>
  1   Alice      25
  2   Bob        30
  3   Charlie    35
  4   David      40
  5   Eve        45
```



```{r}
# Perform right join
result <- df_a %>%
  full_join(df_b, by = "ID")

result
```

A tibble: 7 × 5

```
 ID   Name      Age  City          Salary
<dbl> <chr>    <dbl> <chr>          <dbl>
  1   Alice      25  NA                NA
  2   Bob        30  New York       50000
  3   Charlie    35  NA                NA
  4   David      40  Los Angeles    60000
  5   Eve        45  NA                NA
  6   NA         NA  Chicago        55000
  8   NA         NA  Houston        65000
```



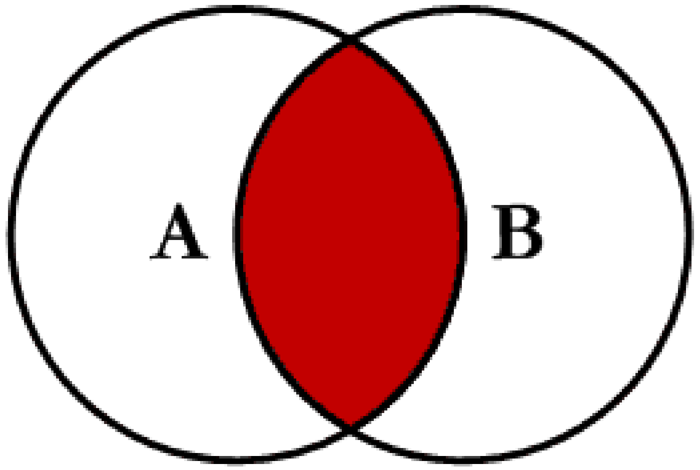
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```

A tibble: 4 × 3

```
 ID   City          Salary
<dbl> <chr>          <dbl>
  2   New York       50000
  4   Los Angeles    60000
  6   Chicago        55000
  8   Houston        65000
```

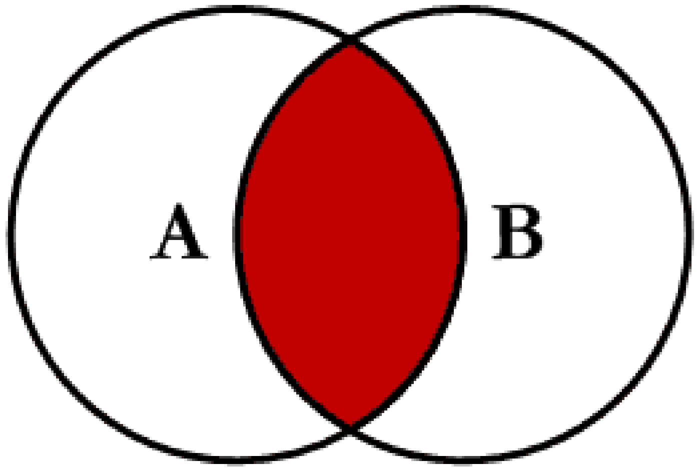
Inner Join

- Keep only data where id is present in both left and right datasets.



Inner Join

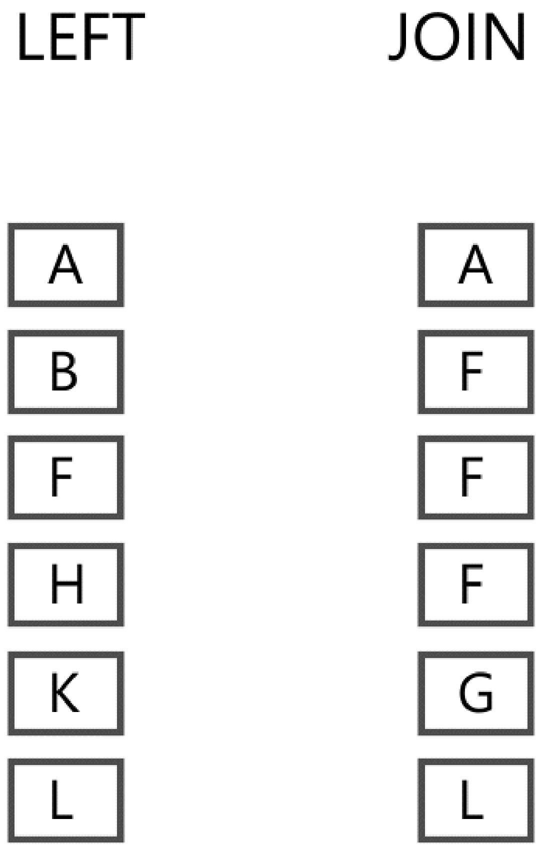
- Keep only data where id is present in both left and right datasets.
- Result has same number of rows as overlapping *join by variable*.



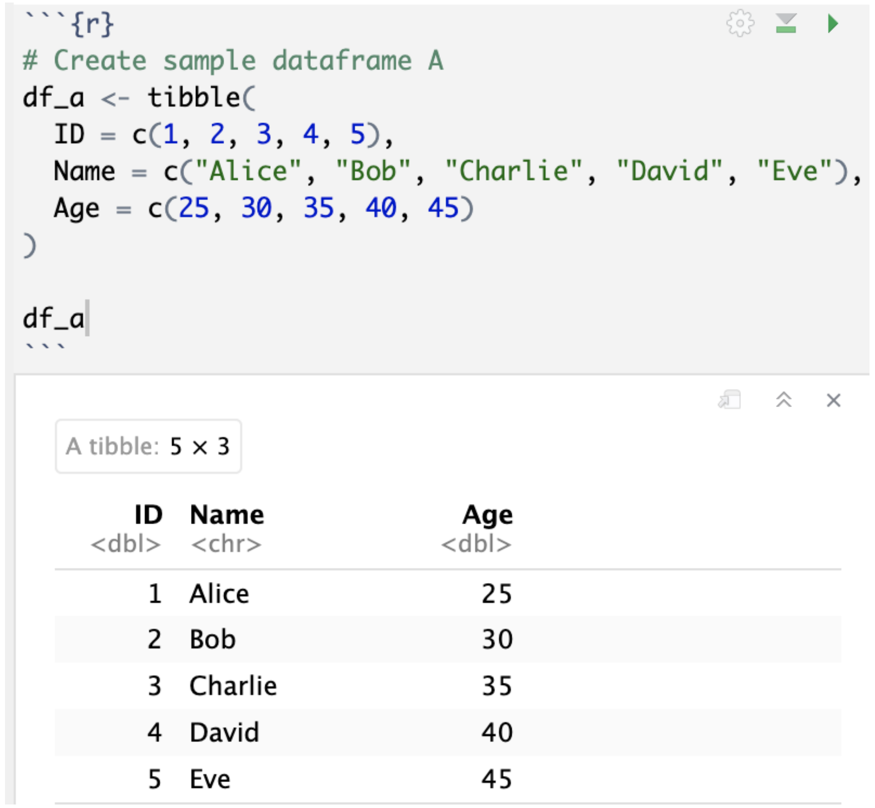
Inner Join

- Keep only data where id is present in both left and right datasets.

- Result has same number of rows as overlapping *join by variable*.



Inner Join (Exercise)

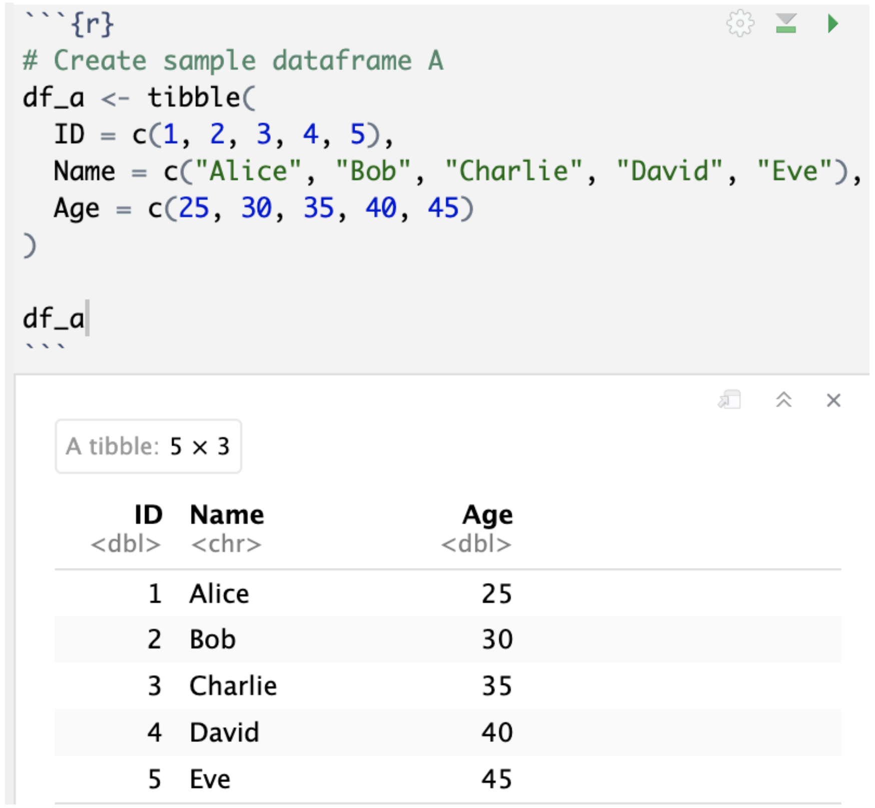


```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```

A tibble: 5 × 3

ID <dbl>	Name <chr>	Age <dbl>
1	Alice	25
2	Bob	30
3	Charlie	35
4	David	40
5	Eve	45

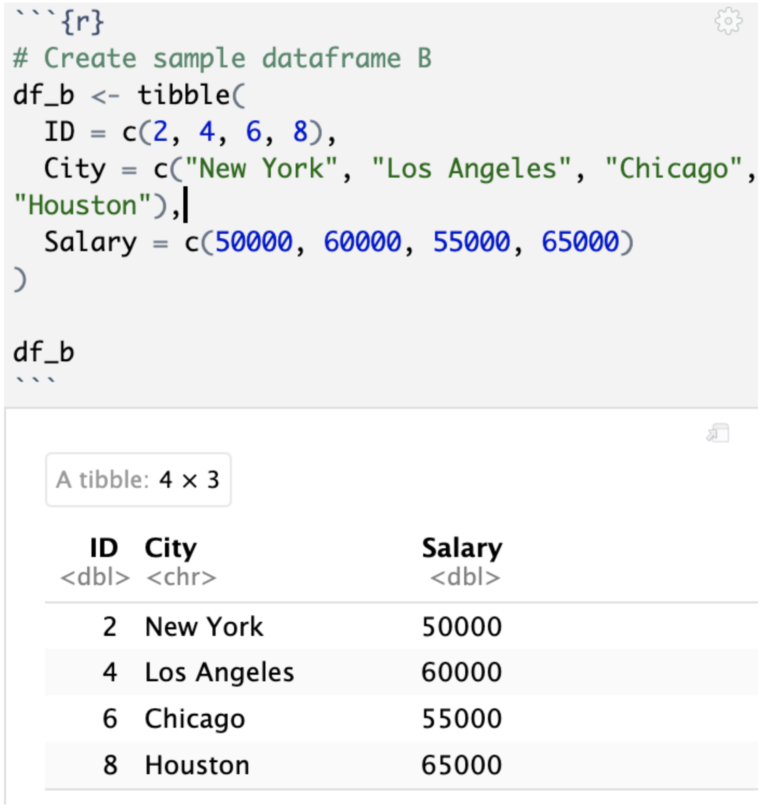
Inner Join (Exercise)



~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

A tibble: 5 × 3

ID <dbl>	Name <chr>	Age <dbl>
1	Alice	25
2	Bob	30
3	Charlie	35
4	David	40
5	Eve	45

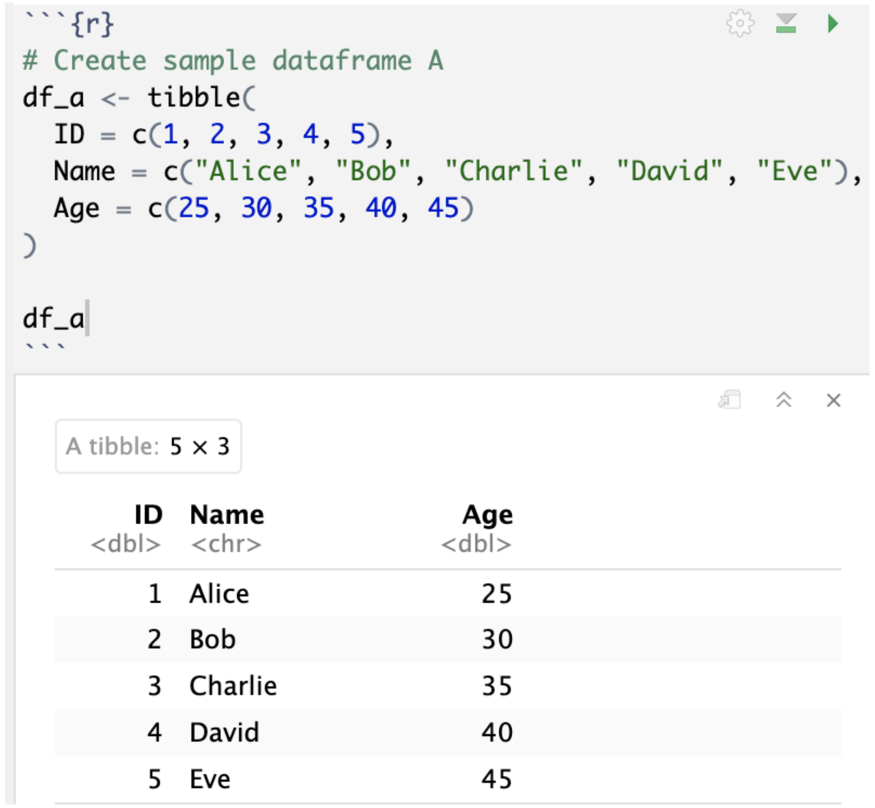


~~~
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```
~~~

A tibble: 4 × 3

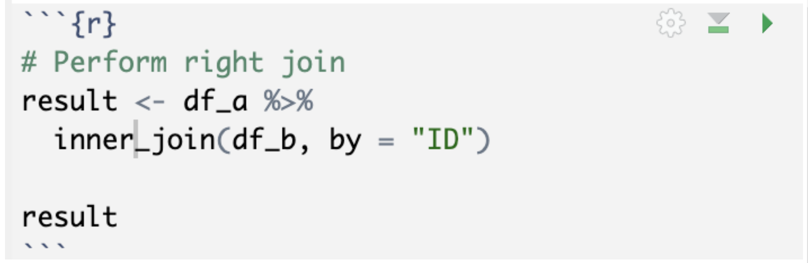
ID <dbl>	City <chr>	Salary <dbl>
2	New York	50000
4	Los Angeles	60000
6	Chicago	55000
8	Houston	65000

Inner Join (Exercise)

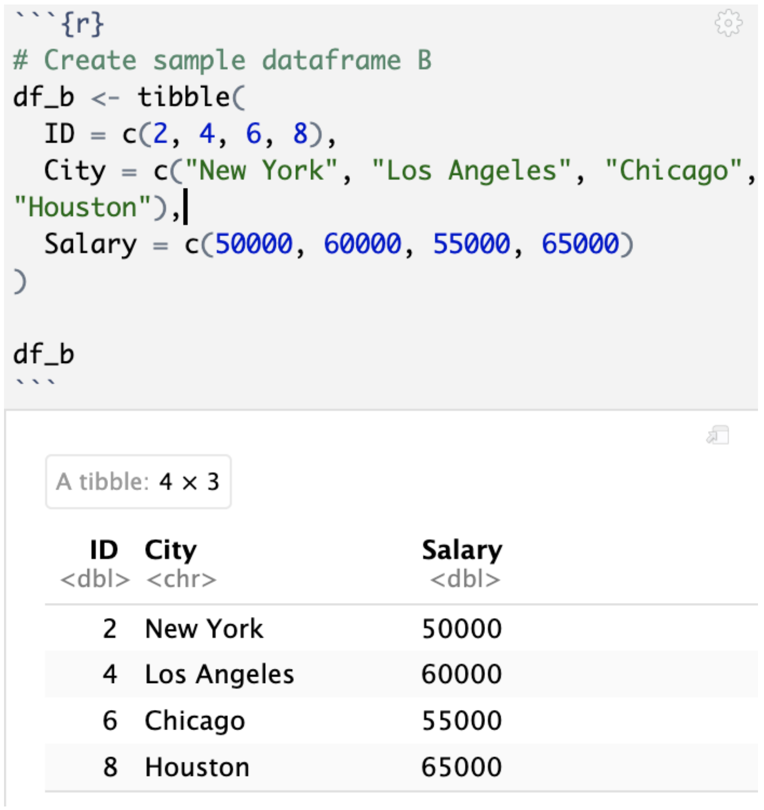


~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

```
A tibble: 5 × 3

   ID  Name       Age
<dbl>  <chr>    <dbl>
    1  Alice       25
    2  Bob         30
    3  Charlie     35
    4  David       40
    5  Eve         45
```



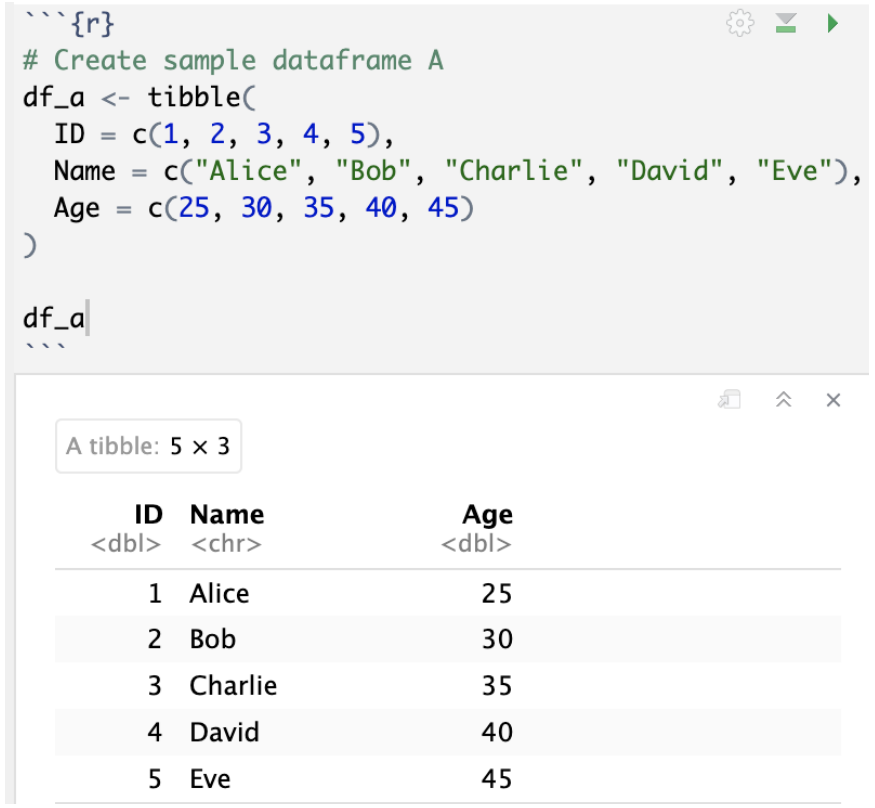
~~~
```{r}
# Perform right join
result <- df_a %>%
  inner_join(df_b, by = "ID")

result
```
~~~



~~~
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```
~~~

```
A tibble: 4 × 3

   ID  City          Salary
<dbl>  <chr>          <dbl>
    2  New York       50000
    4  Los Angeles    60000
    6  Chicago        55000
    8  Houston        65000
```

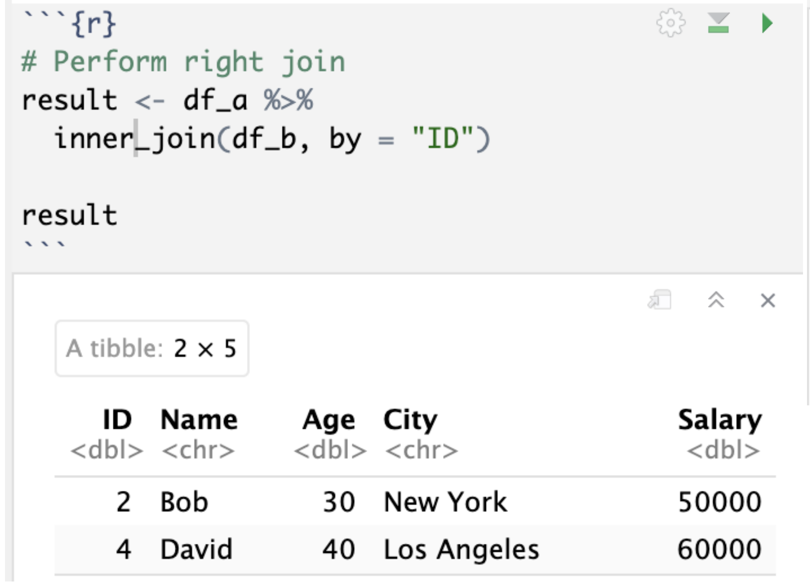
Inner Join (Exercise)



~~~
```{r}
# Create sample dataframe A
df_a <- tibble(
  ID = c(1, 2, 3, 4, 5),
  Name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Age = c(25, 30, 35, 40, 45)
)

df_a
```
~~~

A tibble: 5 × 3

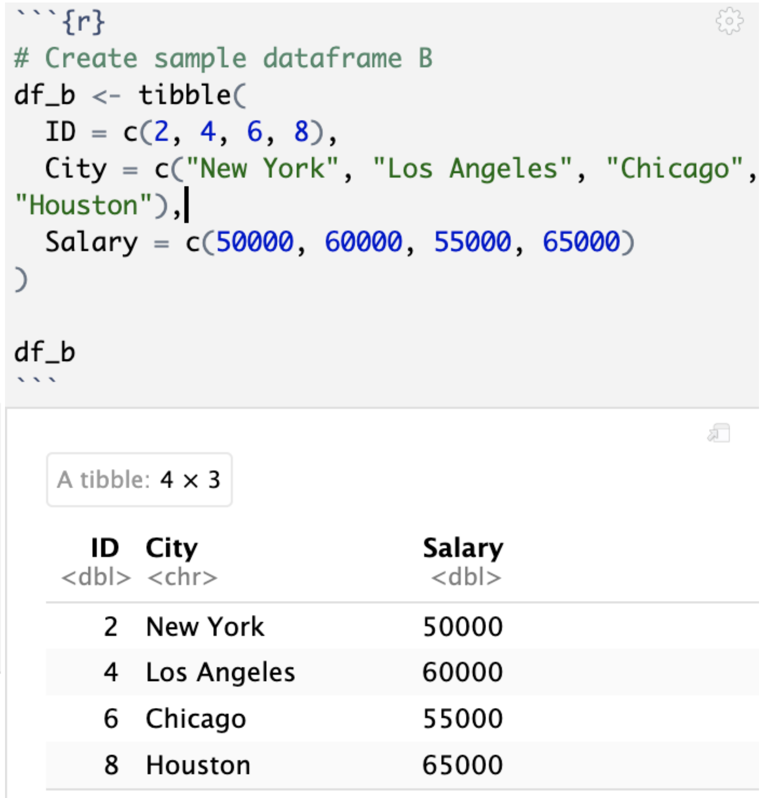
ID <dbl>	Name <chr>	Age <dbl>
1	Alice	25
2	Bob	30
3	Charlie	35
4	David	40
5	Eve	45



~~~
```{r}
# Perform right join
result <- df_a %>%
  inner_join(df_b, by = "ID")

result
```
~~~

A tibble: 2 × 5

ID <dbl>	Name <chr>	Age <dbl>	City <chr>	Salary <dbl>
2	Bob	30	New York	50000
4	David	40	Los Angeles	60000



~~~
```{r}
# Create sample dataframe B
df_b <- tibble(
  ID = c(2, 4, 6, 8),
  City = c("New York", "Los Angeles", "Chicago",
"Houston"),
  Salary = c(50000, 60000, 55000, 65000)
)

df_b
```
~~~

A tibble: 4 × 3

ID <dbl>	City <chr>	Salary <dbl>
2	New York	50000
4	Los Angeles	60000
6	Chicago	55000
8	Houston	65000

Mutating Joins

Mutating Joins

- “mutate” adds ***columns***

Mutating Joins

- “mutate” adds ***columns***
  - After the join, you have variables from both tables in the output.
- number of rows change based on if there duplicates

Mutating Joins

- “mutate” adds ***columns***
    - After the join, you have variables from both tables in the output.

- number of rows change based on if there duplicates

- Examples
    - inner join
    - left_join
    - (right_join)
    - full_join

Filtering joins

- No new columns are added
  - Output always has only columns from left(x) dataset

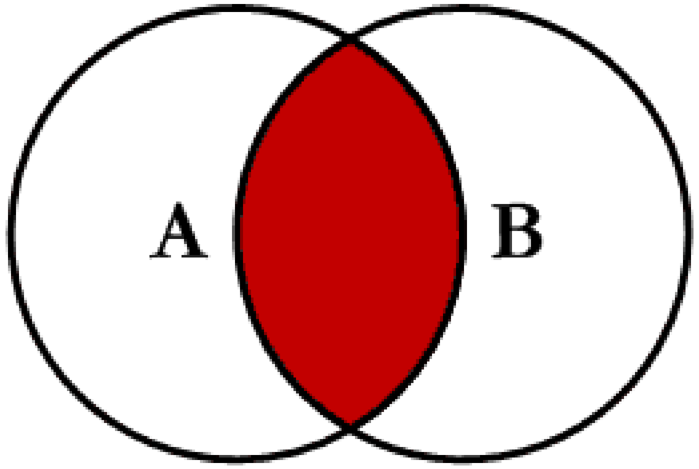
- semi_join
  - Keep rows in **x** that have matches in **y**
  - No duplication if multiple matches in y

Filtering joins

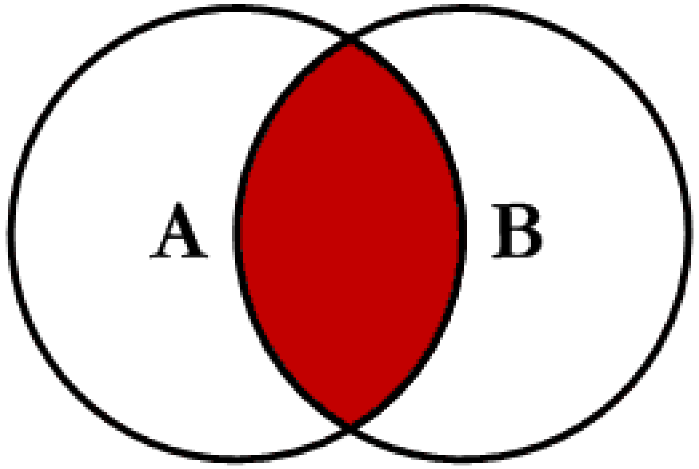
- No new columns are added
  - Output always has only columns from left(x) dataset

- semi_join
  - Keep rows in **x** that have matches in **y**
  - No duplication if multiple matches in y

- semi_join vs inner_join?

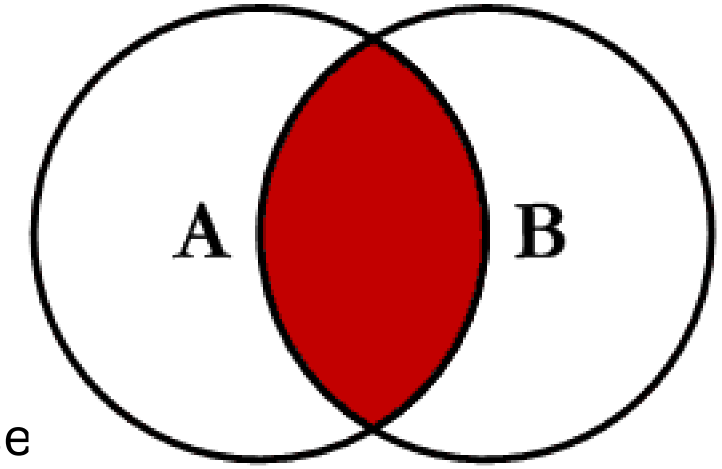


Filtering joins

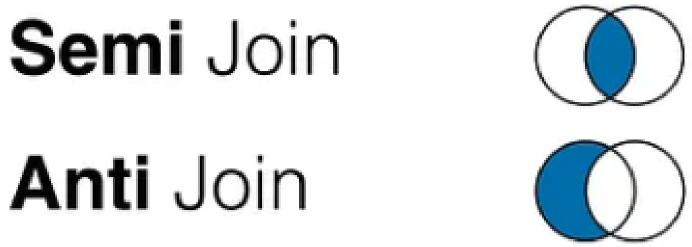


- No new columns are added
    - Output always has only columns from left(x) dataset
- semi_join
    - Keep rows in **x** that have matches in **y**
    - No duplication if multiple matches in y
- anti_join

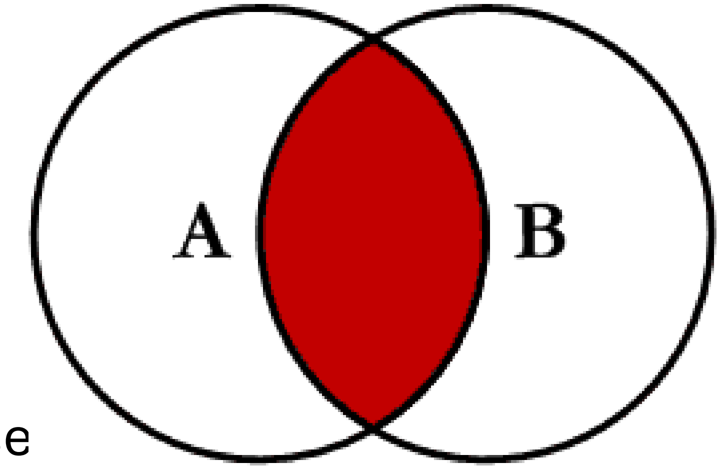
Filtering joins



- No new columns are added
  - Output always has only columns from left(x) datase
- semi_join
  - Keep rows in **x** that have matches in **y**
  - No duplication if multiple matches in y
- anti_join
  - keeps only rows in **x** that **do not** have a match in **y**



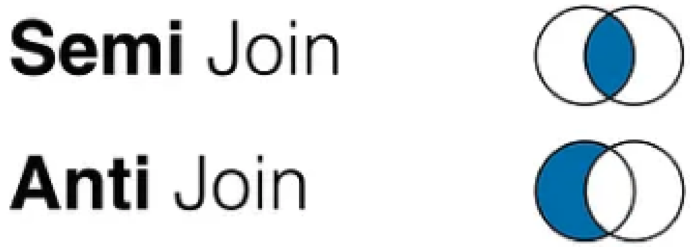
Filtering joins



- No new columns are added
  - Output always has only columns from left(x) datase

- semi_join
  - Keep rows in **x** that have matches in **y**
  - No duplication if multiple matches in y

- anti_join
  - keeps only rows in **x** that **do not** have a match in **y**



Join	**English**
Left join	“Keep all of **x**, add what matches from **y**.”
Right join	“Keep all of **y**, add what matches from **x** (but really just swap and do a left join).”
Inner join	“Keep the overlap of both”
Full join	“Keep the union from both”
Semi join	“Keep the parts of **x** that have a match in **y** — but don’t bring along y columns”
Anti join	“Keep the parts of **x** that have **no** match in **y** — don’t worry about y columns”

Common pitfalls

- join key isn’t actually unique in one or both datasets
- If you don’t specify “by =”
- NAs
    - losing information unintentionally [inner join]
    - unmatched cases get NAs in the new columns [left/full join]
        - *causing later issues*
- forgetting join direction matters

Tips

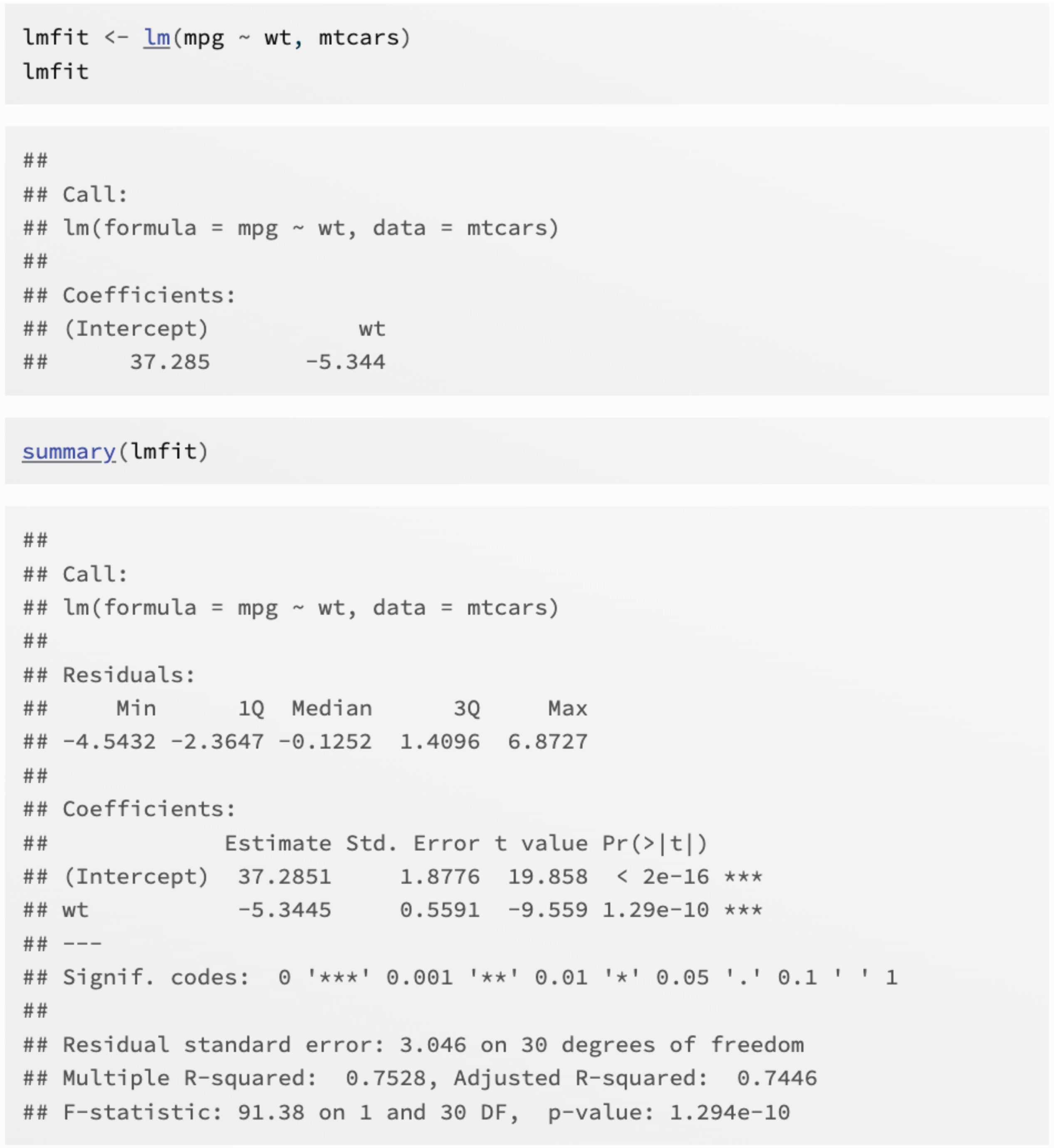
- ensure uniqueness of your key in both tables, and use the appropriate join
- be explicit in “by =”
- validate and check results after joining.

Broom

- Part of tidyverse
- "The broom package takes the messy output of built-in functions in R, such as lm, nls, or t.test, and turns them into tidy tibbles."
- broom provides three verbs to make it convenient to interact with model objects:
  - tidy() summarizes information about model components
  - glance() reports information about the entire model
  - augment() adds informations about observations to a dataset

tidy()

- “constructs a tibble that summarizes the model’s statistical findings. This includes coefficients and p-values for each term in a regression,…”



```r
lmfit <- lm(mpg ~ wt, mtcars)
lmfit
```

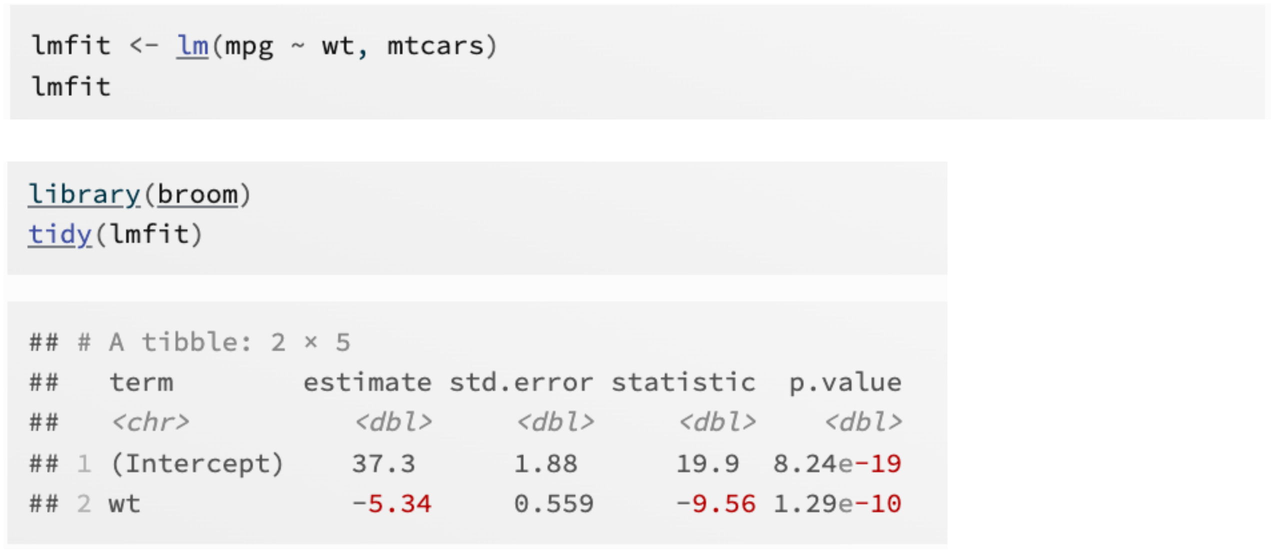
```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

```r
summary(lmfit)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

tidy()

- “constructs a tibble that summarizes the model’s statistical findings. This includes coefficients and p-values for each term in a regression,…”



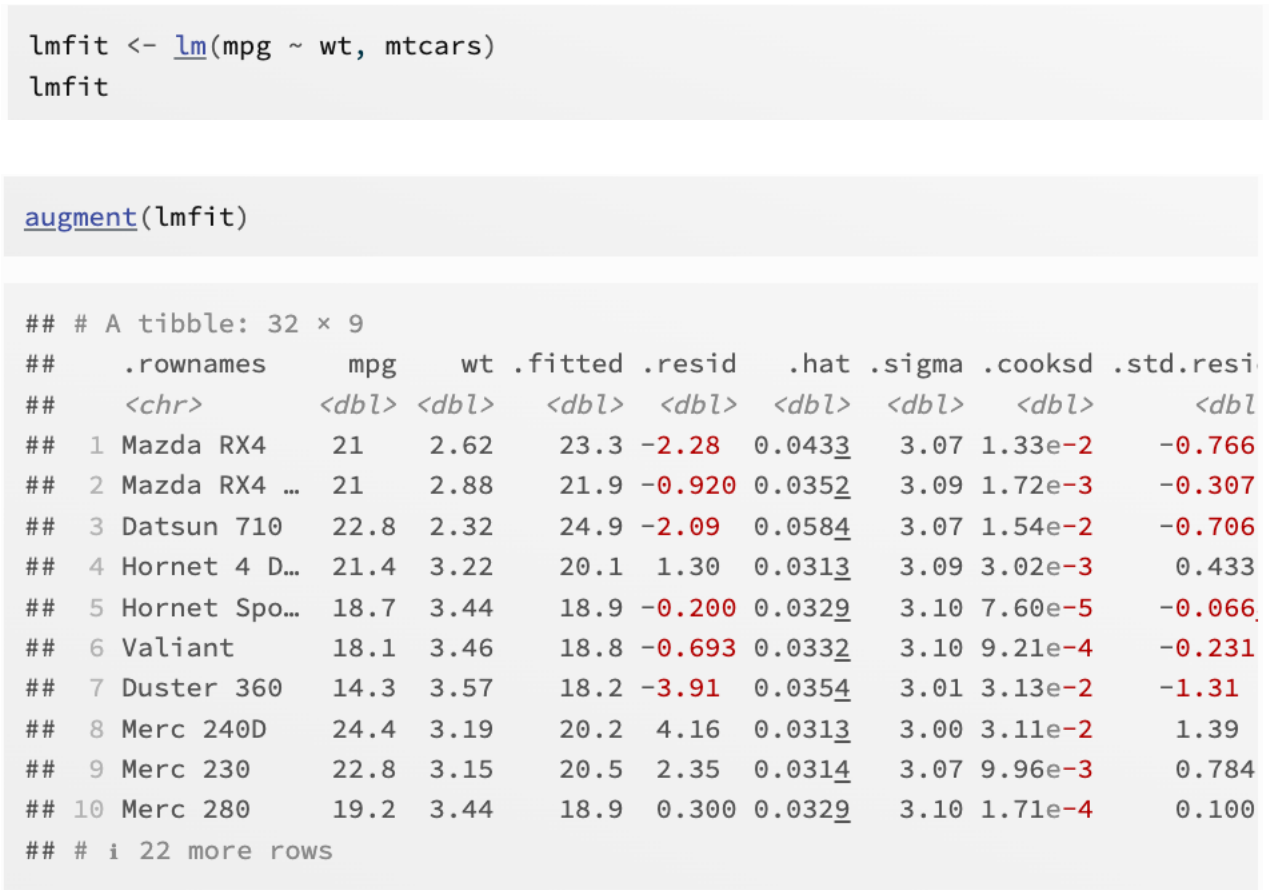
```
lmfit <- lm(mpg ~ wt, mtcars)
lmfit
```

```
library(broom)
tidy(lmfit)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    37.3      1.88      19.9  8.24e-19
## 2 wt             -5.34     0.559     -9.56 1.29e-10
```

augment()

- Instead of viewing the coefficients, you might be interested in the fitted values and residuals for each of the original points in the regression.
- For this, use augment, which augments the original data with information from the model:



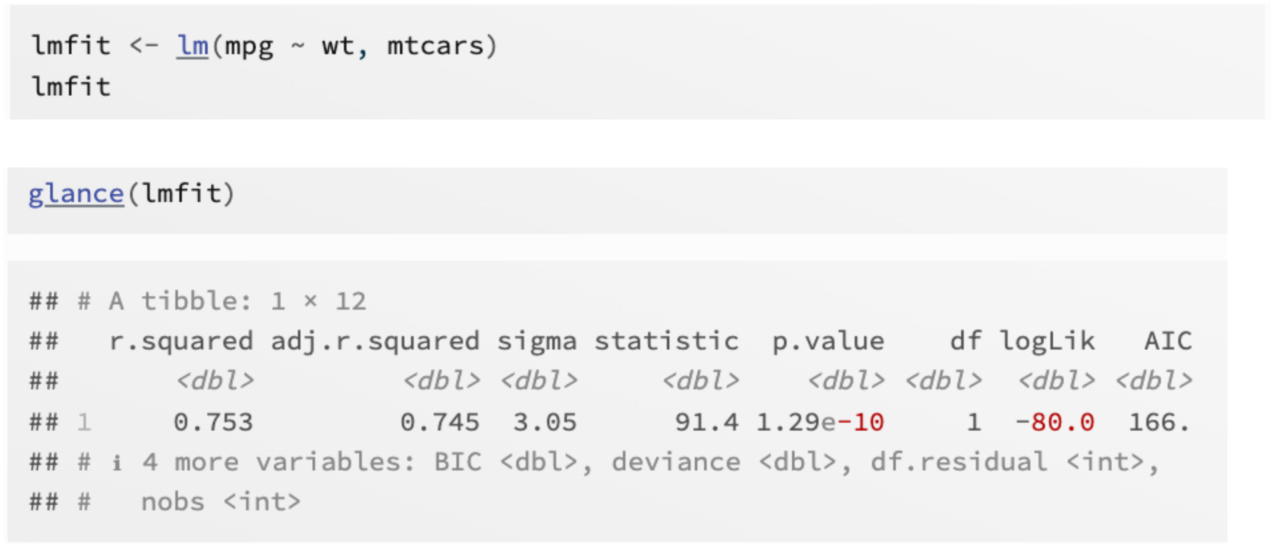
```
lmfit <- lm(mpg ~ wt, mtcars)
lmfit
```

```
augment(lmfit)
```

```
## # A tibble: 32 × 9
##    .rownames      mpg    wt .fitted .resid   .hat .sigma .cooksd .std.resi
##    <chr>        <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>      <dbl
##  1 Mazda RX4     21    2.62    23.3 -2.28  0.0433   3.07 1.33e-2    -0.766
##  2 Mazda RX4 …   21    2.88    21.9 -0.920 0.0352   3.09 1.72e-3    -0.307
##  3 Datsun 710    22.8  2.32    24.9 -2.09  0.0584   3.07 1.54e-2    -0.706
##  4 Hornet 4 D…   21.4  3.22    20.1  1.30  0.0313   3.09 3.02e-3     0.433
##  5 Hornet Spo…   18.7  3.44    18.9 -0.200 0.0329   3.10 7.60e-5    -0.066
##  6 Valiant       18.1  3.46    18.8 -0.693 0.0332   3.10 9.21e-4    -0.231
##  7 Duster 360    14.3  3.57    18.2 -3.91  0.0354   3.01 3.13e-2    -1.31
##  8 Merc 240D     24.4  3.19    20.2  4.16  0.0313   3.00 3.11e-2     1.39
##  9 Merc 230      22.8  3.15    20.5  2.35  0.0314   3.07 9.96e-3     0.784
## 10 Merc 280      19.2  3.44    18.9  0.300 0.0329   3.10 1.71e-4     0.100
## # ℹ 22 more rows
```

glance()

- several summary statistics are computed for the entire regression, such as R^2 and the F-statistic. These can be accessed with the glance function



```r
lmfit <- lm(mpg ~ wt, mtcars)
lmfit
```

```r
glance(lmfit)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl>
## 1     0.753         0.745  3.05      91.4 1.29e-10     1  -80.0  166.
## # ℹ 4 more variables: BIC <dbl>, deviance <dbl>, df.residual <int>,
## #   nobs <int>
```