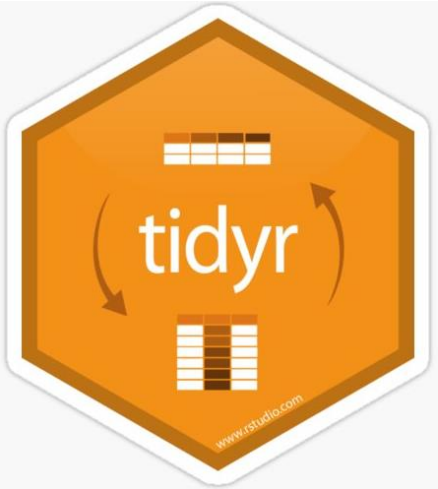
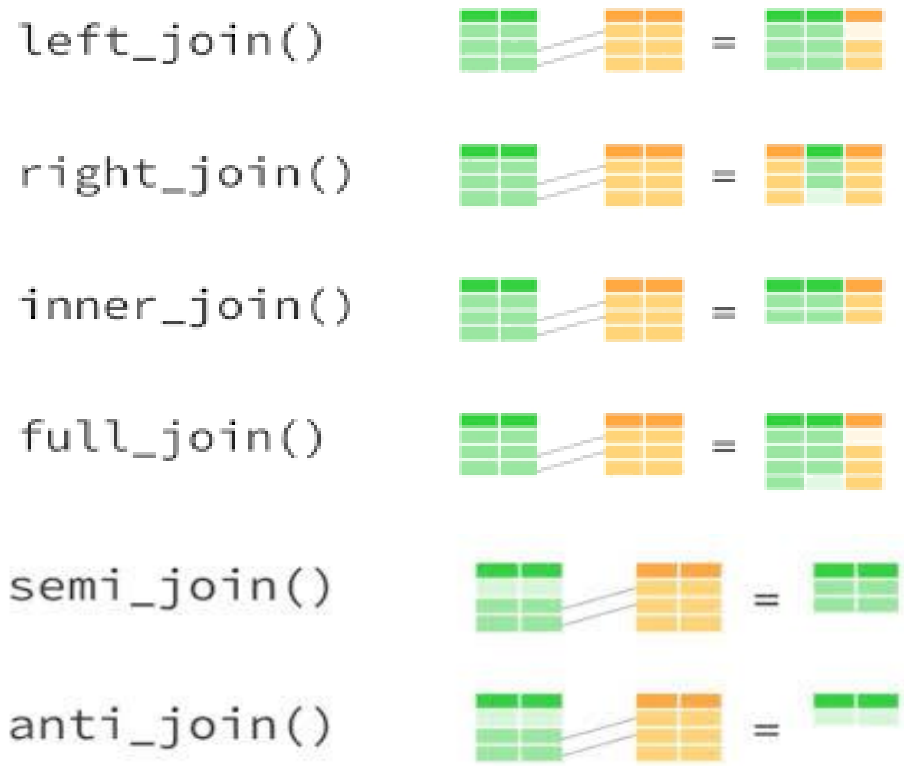
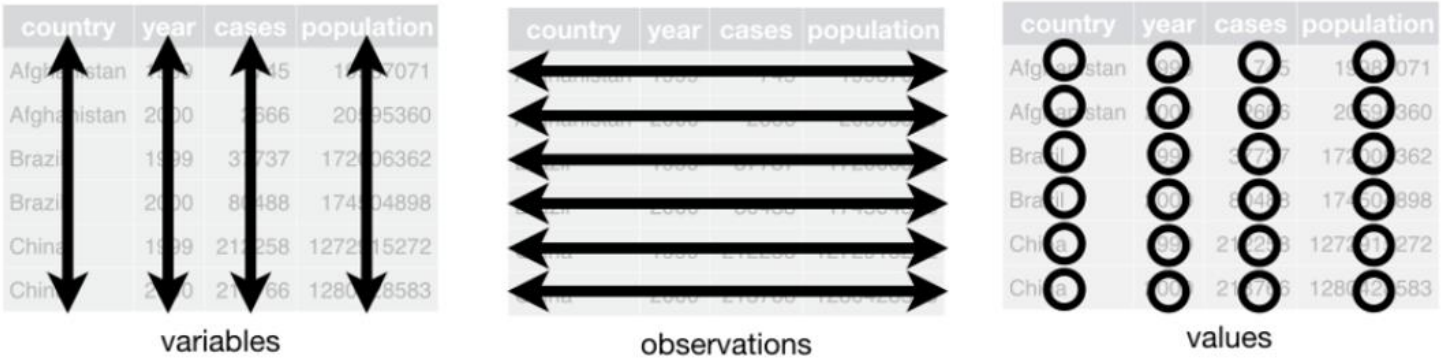
Data Manipulation II

Business Intelligence







Terry College of Business
UNIVERSITY OF GEORGIA

Tidy Data & Transformations

Tidy Data

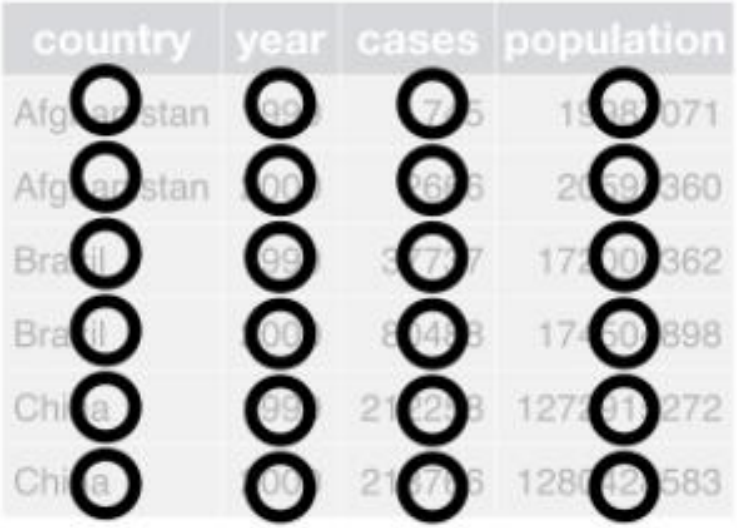
Each **variable** must have **its own column**.

variables

Each **observation** must have **its own row**

observations

Each **value** must have **its own cell**



values

Pivoting

Most untidy datasets:

1. One **observation** scattered across **multiple rows**.

2. One **variable** spread across **multiple columns**.

Pivot wider

An **observation** is scattered across **multiple rows**.

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Pivot wider

```
table2
#> # A tibble: 12 x 4
#>   country      year type           count
#>   <chr>       <int> <chr>          <int>
#> 1 Afghanistan  1999 cases            745
#> 2 Afghanistan  1999 population  19987071
#> 3 Afghanistan  2000 cases           2666
#> 4 Afghanistan  2000 population  20595360
#> 5 Brazil       1999 cases          37737
#> 6 Brazil       1999 population 172006362
#> # … with 6 more rows
```

To identify:

1. The column to take variable names from. Here, it's type.

2. The column to take values from. Here it's count.

Pivot wider

```
table2
#> # A tibble: 12 x 4
#>   country      year type           count
#>   <chr>       <int> <chr>          <int>
#> 1 Afghanistan  1999 cases            745
#> 2 Afghanistan  1999 population  19987071
#> 3 Afghanistan  2000 cases           2666
#> 4 Afghanistan  2000 population  20595360
#> 5 Brazil       1999 cases          37737
#> 6 Brazil       1999 population 172006362
#> # … with 6 more rows
```

```
table2 %>%
  pivot_wider(names_from = type, values_from = count)
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

Pivot longer

Some of the column names are **not names** of variables, but **values** of a variable.

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Pivot longer

```
table4a
#> # A tibble: 3 x 3
#>   country     `1999` `2000`
#> * <chr>        <int>  <int>
#> 1 Afghanistan    745   2666
#> 2 Brazil       37737  80488
#> 3 China       212258 213766
```

To identify:

1. The set of columns whose names are values, not variables.
2. The name of the variable to move the column names to. Here it is year.
3. The name of the variable to move the column values to. Here it's cases.

Pivot longer

```
table4a
#> # A tibble: 3 x 3
#>   country     `1999` `2000`
#> * <chr>        <int>  <int>
#> 1 Afghanistan    745   2666
#> 2 Brazil       37737  80488
#> 3 China       212258 213766
```

```
table4a %>%
  pivot_longer(c(`1999`, `2000`),
               names_to = "year", values_to = "cases")
#> # A tibble: 6 x 3
#>   country     year   cases
#>   <chr>       <chr>  <int>
#> 1 Afghanistan 1999     745
#> 2 Afghanistan 2000    2666
#> 3 Brazil      1999   37737
#> 4 Brazil      2000   80488
#> 5 China       1999  212258
#> 6 China       2000  213766
```

Additional transformations

Other types of “not so common” problems

1. **One column** contains data about **two variables**.

2. **Two columns** contain data about **one variable**.

Separate

One column contains data about **two variables.**

country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

table3

country	year	cases	population
Afghanistan	1999	**745**	19987071
Afghanistan	2000	**2666**	20595360
Brazil	1999	**37737**	172006362
Brazil	2000	**80488**	174504898
China	1999	**212258**	1272915272
China	2000	**213766**	1280428583

Separate

```
table3
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/20595360
#> 3 Brazil       1999 37737/172006362
#> 4 Brazil       2000 80488/174504898
#> 5 China        1999 212258/1272915272
#> 6 China        2000 213766/1280428583
```

To identify:

1. The name of the column to separate.

2. The names of the columns to separate into

Separate

```
table3
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/20595360
#> 3 Brazil       1999 37737/172006362
#> 4 Brazil       2000 80488/174504898
#> 5 China        1999 212258/1272915272
#> 6 China        2000 213766/1280428583
```

```
table3 %>%
  separate(rate, into = c("cases", "population"))
#> # A tibble: 6 x 4
#>   country      year cases  population
#>   <chr>       <int> <chr>  <chr>
#> 1 Afghanistan  1999 745    19987071
#> 2 Afghanistan  2000 2666   20595360
#> 3 Brazil       1999 37737  172006362
#> 4 Brazil       2000 80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

By default, separate() will split values wherever it sees a non-alphanumeric character

```
table3 %>%
  separate(rate, into = c("cases", "population"), sep = "/")
```

Separate

```
table3 %>%
  separate(year, into = c("century", "year"), sep = 2)
#> # A tibble: 6 x 4
#>   country     century year  rate
#>   <chr>       <chr>   <chr> <chr>
#> 1 Afghanistan 19      99    745/19987071
#> 2 Afghanistan 20      00    2666/20595360
#> 3 Brazil      19      99    37737/172006362
#> 4 Brazil      20      00    80488/174504898
#> 5 China       19      99    212258/1272915272
#> 6 China       20      00    213766/1280428583
```

Unite

Two columns

contain data about

one variable.

country	year	rate
Afghanistan	1999	745 / 19987071
Afghanistan	2000	2666 / 20595360
Brazil	1999	37737 / 172006362
Brazil	2000	80488 / 174504898
China	1999	212258 / 1272915272
China	2000	213766 / 1280428583

country	century	year	rate
Afghanistan	19	99	745 / 19987071
Afghanistan	20	0	2666 / 20595360
Brazil	19	99	37737 / 172006362
Brazil	20	0	80488 / 174504898
China	19	99	212258 / 1272915272
China	20	0	213766 / 1280428583

table6

To identify

1. The name of the new variable to create.
2. A set of columns to combine.

Unite

To identify

1. The name of the new variable to create.
2. A set of columns to combine.

```
table5 %>%
  unite(new, century, year)
#> # A tibble: 6 x 3
#>   country     new   rate
#>   <chr>       <chr> <chr>
#> 1 Afghanistan 19_99 745/19987071
#> 2 Afghanistan 20_00 2666/20595360
#> 3 Brazil      19_99 37737/172006362
#> 4 Brazil      20_00 80488/174504898
#> 5 China       19_99 212258/1272915272
#> 6 China       20_00 213766/1280428583
```

Unite

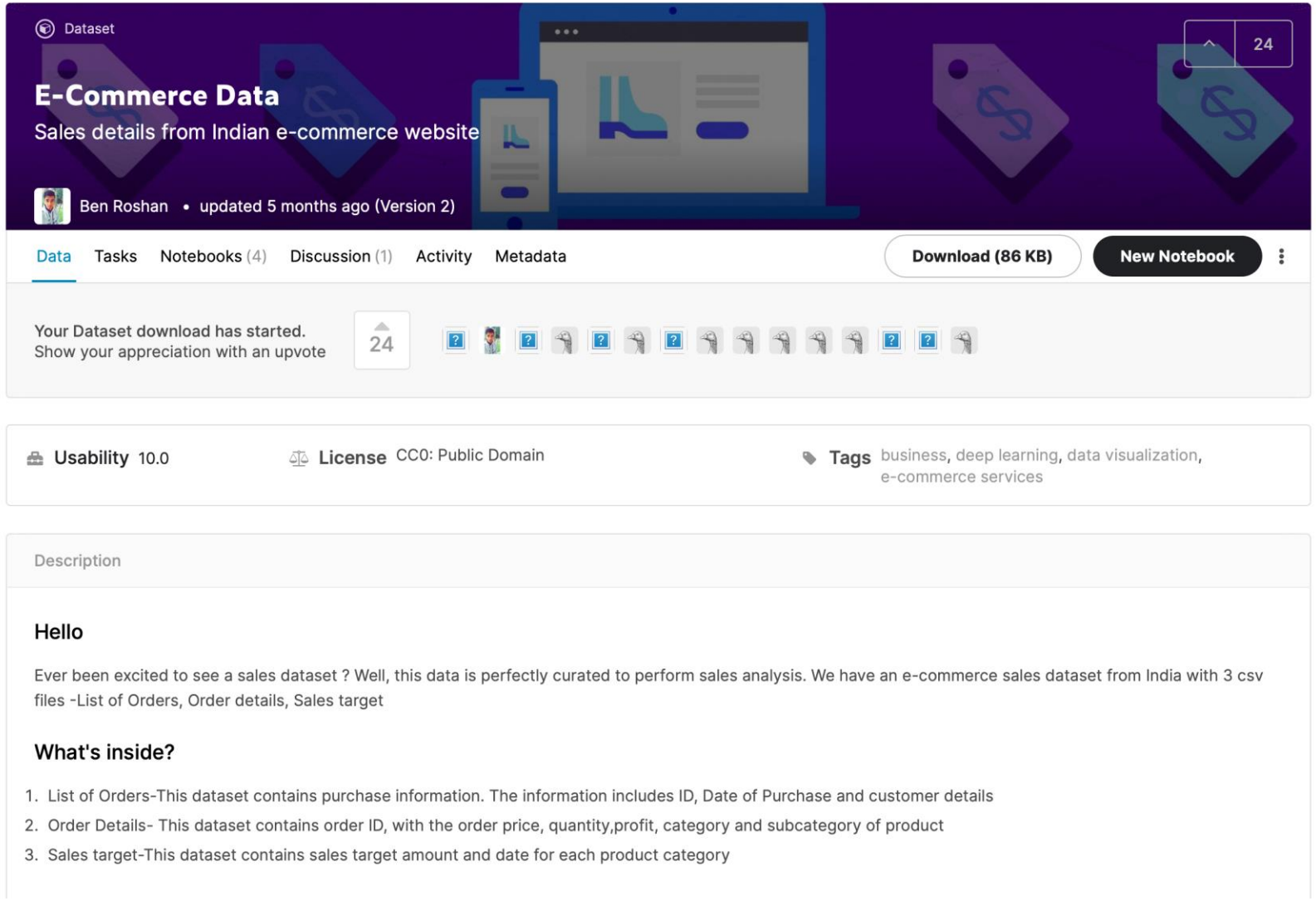
By default, unite() will place an underscore (_) between the values from different columns.

```
table5 %>%
  unite(new, century, year, sep = "")
#> # A tibble: 6 x 3
#>   country     new   rate
#>   <chr>       <chr> <chr>
#> 1 Afghanistan 1999  745/19987071
#> 2 Afghanistan 2000  2666/20595360
#> 3 Brazil      1999  37737/172006362
#> 4 Brazil      2000  80488/174504898
#> 5 China       1999  212258/1272915272
#> 6 China       2000  213766/1280428583
```

Relational Data - Joins

Data

Rely on e-commerce data from Kaggle



https://www.kaggle.com/benroshan/ecommerce-data?select=Order+Details.csv

Data Import

```
library(tidyverse)

orderList <- read_csv("List of Orders.csv")
orderDetails <- read_csv("Order Details.csv")
salesTarget <- read_csv("Sales target.csv")
```

https://www.kaggle.com/benroshan/ecommerce-data?select=Order+Details.csv

orderList Table

```
orderList
# A tibble: 500 x 5
   `Order ID` `Order Date` CustomerName State            City
   <chr>      <chr>        <chr>        <chr>            <chr>
 1 B-25601    01-04-2018   Bharat       Gujarat          Ahmedabad
 2 B-25602    01-04-2018   Pearl        Maharashtra      Pune
 3 B-25603    03-04-2018   Jahan        Madhya Pradesh   Bhopal
 4 B-25604    03-04-2018   Divsha       Rajasthan        Jaipur
 5 B-25605    05-04-2018   Kasheen      West Bengal      Kolkata
 6 B-25606    06-04-2018   Hazel        Karnataka        Bangalore
 7 B-25607    06-04-2018   Sonakshi     Jammu and Kashmir Kashmir
 8 B-25608    08-04-2018   Aarushi      Tamil Nadu       Chennai
 9 B-25609    09-04-2018   Jitesh       Uttar Pradesh    Lucknow
10 B-25610    09-04-2018   Yogesh       Bihar            Patna
# … with 490 more rows
```


https://www.kaggle.com/benroshan/ecommerce-data?select=Order+Details.csv

orderDetails Table

```
orderDetails
# A tibble: 1,500 x 6
   `Order ID` Amount Profit Quantity Category    `Sub-Category`
   <chr>       <dbl>  <dbl>    <dbl> <chr>       <chr>
 1 B-25601      1275  -1148        7 Furniture   Bookcases
 2 B-25601        66    -12        5 Clothing    Stole
 3 B-25601         8     -2        3 Clothing    Hankerchief
 4 B-25601        80    -56        4 Electronics Electronic Games
 5 B-25602       168   -111        2 Electronics Phones
 6 B-25602       424   -272        5 Electronics Phones
 7 B-25602      2617   1151        4 Electronics Phones
 8 B-25602       561    212        3 Clothing    Saree
 9 B-25602       119     -5        8 Clothing    Saree
10 B-25603      1355    -60        5 Clothing    Trousers
# … with 1,490 more rows
```

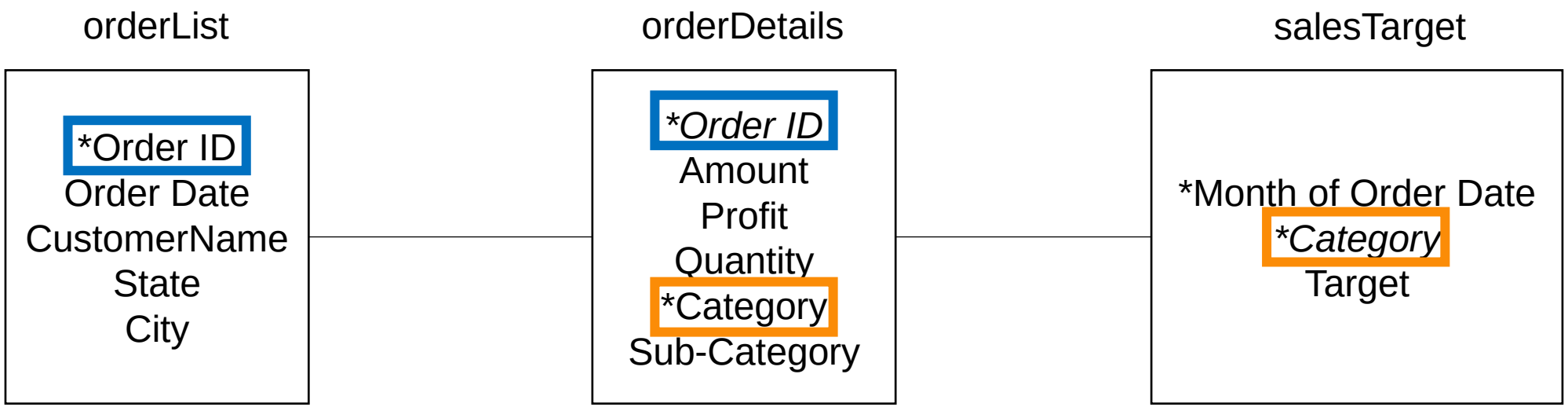

https://www.kaggle.com/benroshan/ecommerce-data?select=Order+Details.csv

salesTarget Table

```
salesTarget
# A tibble: 36 x 3
   `Month of Order Date` Category  Target
   <chr>                 <chr>      <dbl>
 1 Apr-18                Furniture  10400
 2 May-18                Furniture  10500
 3 Jun-18                Furniture  10600
 4 Jul-18                Furniture  10800
 5 Aug-18                Furniture  10900
 6 Sep-18                Furniture  11000
 7 Oct-18                Furniture  11100
 8 Nov-18                Furniture  11300
 9 Dec-18                Furniture  11400
10 Jan-19                Furniture  11500
# … with 26 more rows
```

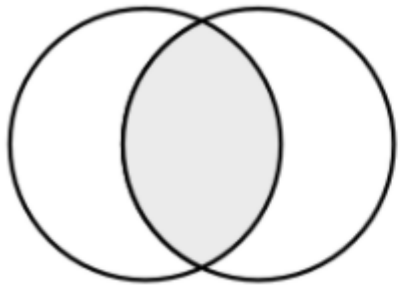

https://www.kaggle.com/benroshan/ecommerce-data?select=Order+Details.csv

The Data


orderList
*Order ID
Order Date
CustomerName
State
City
orderDetails
*Order ID
Amount
Profit
Quantity
*Category
Sub-Category
salesTarget
*Month of Order Date
*Category
Target

Inner Join

Return all rows with matching values in **orderDetails** and **orderList**, including all columns from **orderDetails** and **orderList**.

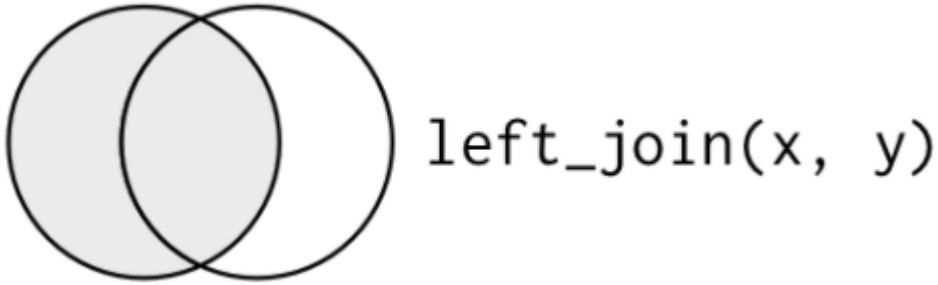


inner_join(x, y)


```
inner_join(orderDetails, orderList, by = "Order ID")

# A tibble: 1,500 x 10
   `Order ID` Amount Profit Quantity Category `Sub-Category` `Order Date`
   <chr>       <dbl>  <dbl>    <dbl> <chr>    <chr>          <chr>
 1 B-25601      1275  -1148        7 Furnitu… Bookcases      01-04-2018
 2 B-25601        66    -12        5 Clothing Stole          01-04-2018
 3 B-25601         8     -2        3 Clothing Hankerchief    01-04-2018
 4 B-25601        80    -56        4 Electro… Electronic Ga… 01-04-2018
 5 B-25602       168   -111        2 Electro… Phones         01-04-2018
 6 B-25602       424   -272        5 Electro… Phones         01-04-2018
 7 B-25602      2617   1151        4 Electro… Phones         01-04-2018
 8 B-25602       561    212        3 Clothing Saree          01-04-2018
 9 B-25602       119     -5        8 Clothing Saree          01-04-2018
10 B-25603      1355    -60        5 Clothing Trousers       03-04-2018
# … with 1,490 more rows, and 3 more variables: CustomerName <chr>, State <chr>,
#   City <chr>
```

Left Join

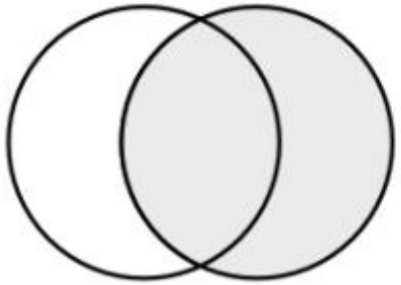
Returns all rows and columns from **orderDetails** and extends them by adding all columns from **orderList**.



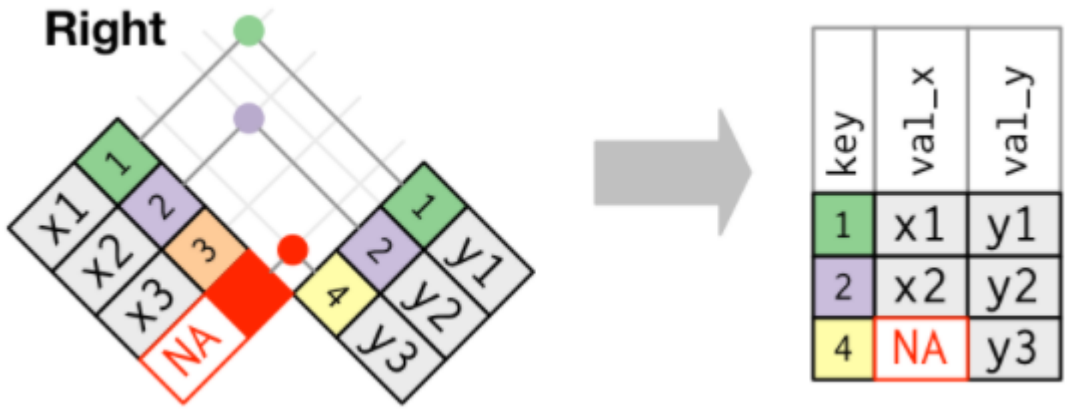

```
left_join(orderDetails, orderList, by = "Order ID")

# A tibble: 1,500 x 10
   `Order ID` Amount Profit Quantity Category `Sub-Category` `Order Date`
   <chr>       <dbl>  <dbl>    <dbl> <chr>    <chr>          <chr>
 1 B-25601      1275  -1148        7 Furnitu… Bookcases      01-04-2018
 2 B-25601        66    -12        5 Clothing Stole          01-04-2018
 3 B-25601         8     -2        3 Clothing Hankerchief    01-04-2018
 4 B-25601        80    -56        4 Electro… Electronic Ga… 01-04-2018
 5 B-25602       168   -111        2 Electro… Phones         01-04-2018
 6 B-25602       424   -272        5 Electro… Phones         01-04-2018
 7 B-25602      2617   1151        4 Electro… Phones         01-04-2018
 8 B-25602       561    212        3 Clothing Saree          01-04-2018
 9 B-25602       119     -5        8 Clothing Saree          01-04-2018
10 B-25603      1355    -60        5 Clothing Trousers       03-04-2018
# … with 1,490 more rows, and 3 more variables: CustomerName <chr>, State <chr>,
#   City <chr>
```

Right Join

Returns all rows and columns from **orderList** and extends them by adding all columns from **orderDetails**.



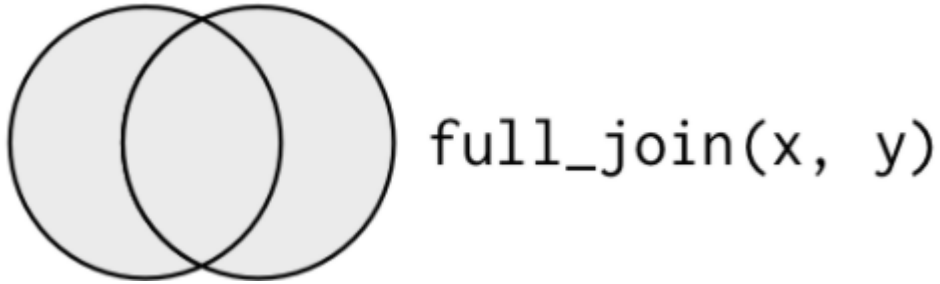
right_join(x, y)

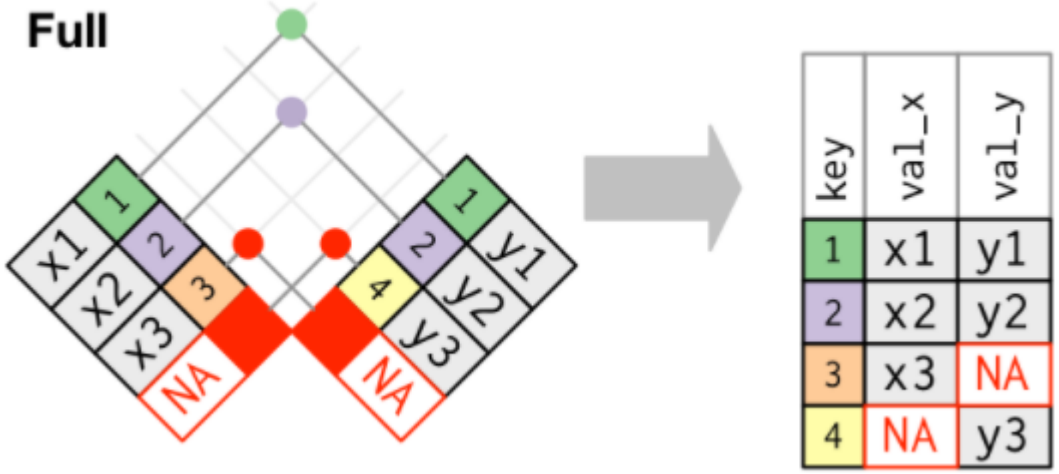



```
right_join(orderDetails, orderList, by = "Order ID")

# A tibble: 1,560 x 10
   `Order ID` Amount Profit Quantity Category `Sub-Category` `Order Date`
   <chr>       <dbl>  <dbl>    <dbl> <chr>    <chr>          <chr>
 1 B-25601      1275  -1148        7 Furnitu… Bookcases      01-04-2018
 2 B-25601        66    -12        5 Clothing Stole          01-04-2018
 3 B-25601         8     -2        3 Clothing Hankerchief    01-04-2018
 4 B-25601        80    -56        4 Electro… Electronic Ga… 01-04-2018
 5 B-25602       168   -111        2 Electro… Phones         01-04-2018
 6 B-25602       424   -272        5 Electro… Phones         01-04-2018
 7 B-25602      2617   1151        4 Electro… Phones         01-04-2018
 8 B-25602       561    212        3 Clothing Saree          01-04-2018
 9 B-25602       119     -5        8 Clothing Saree          01-04-2018
10 B-25603      1355    -60        5 Clothing Trousers       03-04-2018
# … with 1,550 more rows, and 3 more variables: CustomerName <chr>, State <chr>,
#   City <chr>
```

Full Join

Returns all rows and all columns from **orderDetails** and **orderList**






```
full_join(orderDetails, orderList, by = "Order ID")

# A tibble: 1,560 x 10
   `Order ID` Amount Profit Quantity Category `Sub-Category` `Order Date`
   <chr>       <dbl>  <dbl>    <dbl> <chr>    <chr>          <chr>
 1 B-25601      1275  -1148        7 Furnitu… Bookcases      01-04-2018
 2 B-25601        66    -12        5 Clothing Stole          01-04-2018
 3 B-25601         8     -2        3 Clothing Hankerchief    01-04-2018
 4 B-25601        80    -56        4 Electro… Electronic Ga… 01-04-2018
 5 B-25602       168   -111        2 Electro… Phones         01-04-2018
 6 B-25602       424   -272        5 Electro… Phones         01-04-2018
 7 B-25602      2617   1151        4 Electro… Phones         01-04-2018
 8 B-25602       561    212        3 Clothing Saree          01-04-2018
 9 B-25602       119     -5        8 Clothing Saree          01-04-2018
10 B-25603      1355    -60        5 Clothing Trousers       03-04-2018
# … with 1,550 more rows, and 3 more variables: CustomerName <chr>, State <chr>,
#   City <chr>
```

Semi Join

Returns all rows from **orderDetails** where there are matching values in **orderList**, keeping just columns from **orderDetails**

columns

```
semi_join(orderDetails, orderList, by = "Order ID")
```

```
# A tibble: 1,500 x 6
   `Order ID` Amount Profit Quantity Category    `Sub-Category`
   <chr>       <dbl>  <dbl>    <dbl> <chr>       <chr>
 1 B-25601      1275  -1148        7 Furniture   Bookcases
 2 B-25601        66    -12        5 Clothing    Stole
 3 B-25601         8     -2        3 Clothing    Hankerchief
 4 B-25601        80    -56        4 Electronics Electronic Games
 5 B-25602       168   -111        2 Electronics Phones
 6 B-25602       424   -272        5 Electronics Phones
 7 B-25602      2617   1151        4 Electronics Phones
 8 B-25602       561    212        3 Clothing    Saree
 9 B-25602       119     -5        8 Clothing    Saree
10 B-25603      1355    -60        5 Clothing    Trousers
# … with 1,490 more rows
```

Anti Join

Returns all rows from **orderDetails** where there are not matching values in **orderList**, keeping just columns from **orderDetails**


```
anti_join(orderDetails, orderList, by = "Order ID")
# A tibble: 0 x 6
# … with 6 variables: `Order ID` <chr>, Amount <dbl>, Profit <dbl>,
#   Quantity <dbl>, Category <chr>, `Sub-Category` <chr>
```

```
anti_join(orderList, orderDetails, by = "Order ID")
# A tibble: 60 x 5
   `Order ID` `Order Date` CustomerName State City
   <chr>      <chr>        <chr>        <chr> <chr>
 1 NA         NA           NA           NA    NA
 2 NA         NA           NA           NA    NA
 3 NA         NA           NA           NA    NA
 4 NA         NA           NA           NA    NA
 5 NA         NA           NA           NA    NA
 6 NA         NA           NA           NA    NA
 7 NA         NA           NA           NA    NA
 8 NA         NA           NA           NA    NA
 9 NA         NA           NA           NA    NA
10 NA         NA           NA           NA    NA
# ... with 50 more rows
```

What if the join variables have different names?

```
df <- rename(orderList, orderID = `Order ID`) # Creating the problem on purpose
inner_join(orderDetails, df, by = c("Order ID" = "orderID"))

# A tibble: 1,500 x 10
   `Order ID` Amount Profit Quantity Category `Sub-Category` `Order Date`
   <chr>       <dbl>  <dbl>    <dbl> <chr>    <chr>          <chr>
 1 B-25601      1275  -1148        7 Furnitu… Bookcases      01-04-2018
 2 B-25601        66    -12        5 Clothing Stole          01-04-2018
 3 B-25601         8     -2        3 Clothing Hankerchief    01-04-2018
 4 B-25601        80    -56        4 Electro… Electronic Ga… 01-04-2018
 5 B-25602       168   -111        2 Electro… Phones         01-04-2018
 6 B-25602       424   -272        5 Electro… Phones         01-04-2018
 7 B-25602      2617   1151        4 Electro… Phones         01-04-2018
 8 B-25602       561    212        3 Clothing Saree          01-04-2018
 9 B-25602       119     -5        8 Clothing Saree          01-04-2018
10 B-25603      1355    -60        5 Clothing Trousers       03-04-2018
# … with 1,490 more rows, and 3 more variables: CustomerName <chr>, State <chr>,
#   City <chr>
```

At-Home Exercises

Copy and paste the code from the slides into an R Markdown document. Execute the code in R, line by line. Are your results like the ones in the slides? If yes, then write joins to answer the following questions:

Returns all rows from **orderDetails**, and all columns from **orderDetails** and **salesTarget**.

Returns all rows with matching values in **salesTarget** and **orderDetails**, and columns from **orderDetails** only.

Assign all rows with matching values in **salesTarget** and **orderDetails**, and all columns from both tables to a table called **orderSales**. Then return all rows with matching values in **orderSales** and **orderList**, and all columns from both tables.