

***Rotman***

# INTRO TO R – DATA WRANGLING

R Workshop - 2

March 8, 2023 Prepared by Jay Cao / TDMDAL

Website: <https://tdmdal.github.io/r-workshop-202223-winter/>



Rotman School of Management  
UNIVERSITY OF TORONTO

# Plan

- **Tidy Data** (a way to organize data)
- Data manipulation (a continuation)
  - `summarise()` and `group_by()`
  - `_join()` datasets
- We will focus on basics, underlying principles, and best practices

*“Data Scientists spend up to 80% of the time on data cleaning and 20 percent of their time on actual data analysis”*

-- ??????

# Tidy Data – Motivation (FANG Revenue Data)

- Q1. Is the below table organized well for easy analysis?
- Q2. In general, what's a good way to organize/structure tabular data?

```
# A tibble: 4 x 6
```

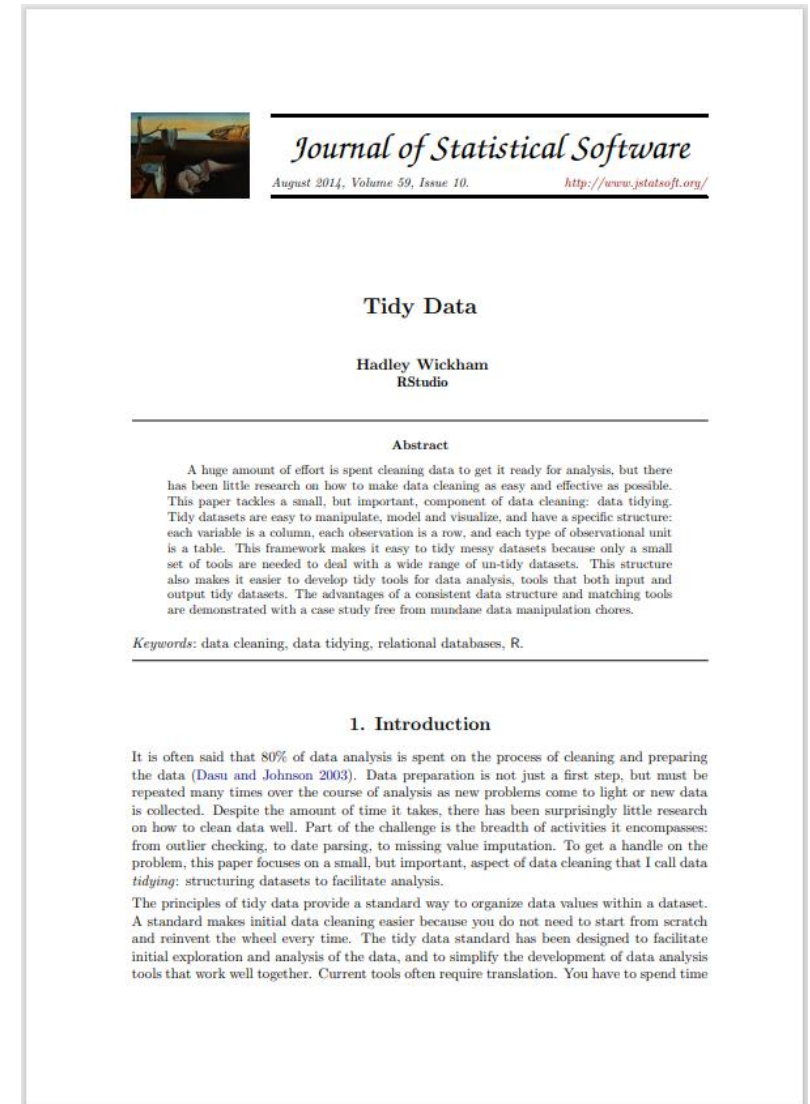
	<b>Ticker</b>	<b>Fiscal Year</b>	<b>Q1</b>	<b>Q2</b>	<b>Q3</b>	<b>Q4</b>
	<i>&lt;chr&gt;</i>	<i>&lt;int&gt;</i>	<i>&lt;dbl&gt;</i>	<i>&lt;dbl&gt;</i>	<i>&lt;dbl&gt;</i>	<i>&lt;dbl&gt;</i>
1	AMZN	2018	51.0	52.9	56.6	72.4
2	FB	2018	12.0	13.2	13.7	16.9
3	GOOG	2018	31.1	32.7	33.7	39.3
4	NFLX	2018	3.70	3.91	4.00	4.19

# Tidy Data

“If you have a consistent data structure, it's easier to learn the tools that work with it because they have an underlying uniformity.”

-- R for Data Science (Chapter 12)

- A (**One**) way to organize **tabular** data
- Definition
  - Each **variable** forms a **column**.
  - Each **observation**, or **case**, forms a **row**.
  - Each **type of observational unit** forms a **table**
- Why tidy data?
  - A great way to organize data for maintainability
  - Once in tidy data, it's easy to use the toolset from *tidyverse* to manipulate them



# Messy Data – Example 1

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

# Messy Data – Example 1 / Why is it Messy?

- Values as column names problem
- Hard to retrieve data and analyze them in a consistent way
  - **how many treatments in total**
  - get average result by person
  - get average result by treatment
  - get overall average result

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

# Messy Data – Example 1 / Why is it Messy?

- Values as column names problem
- Hard to retrieve data and analyze them in a consistent way
  - how many treatments in total
  - **get average result by person**
  - get average result by treatment
  - get overall average result

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

# Messy Data – Example 1 / Why is it Messy?

- Values as column names problem
- Hard to retrieve data and analyze them in a consistent way
  - how many treatments in total
  - get average result by person
  - **get average result by treatment**
  - get overall average result

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.



# Messy Data – Example 1 / Why is it Messy?

- Values as column names problem
- Hard to retrieve data and analyze them in a consistent way
  - how many treatments in total
  - get average result by person
  - get average result by treatment
  - **get overall average result**

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

## Messy Data – Example 2

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Table 2: The same data as in Table 1 but structured differently.

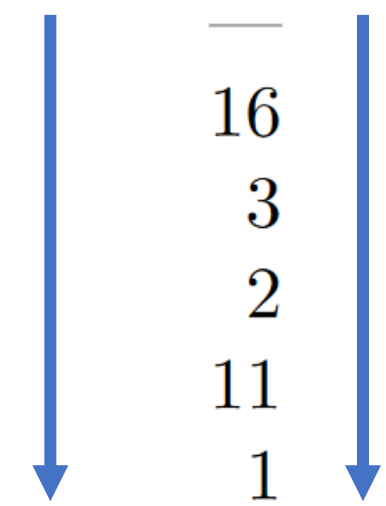
# The Tidy Version

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

# The Tidy Version – Why is it Tidy

- All column-wise operations
  - how many treatments in total
  - get average result by person
  - get average result by treatment
  - get overall average result

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1



# Back to the FANG Revenue Data

```
# A tibble: 4 x 6
```

```
  Ticker `Fiscal Year`    Q1    Q2    Q3    Q4
  <chr>      <int> <dbl> <dbl> <dbl> <dbl>
1 AMZN      2018  51.0  52.9  56.6  72.4
2 FB        2018  12.0  13.2  13.7  16.9
3 GOOG      2018  31.1  32.7  33.7  39.3
4 NFLX      2018   3.70  3.91  4.00  4.19
```

# From Messy to Tidy (One Example)

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.



name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

# `pivot_longer()` from tidyr package

```
# A tibble: 3 x 3
```

```
  name          treatmenta treatmentb
  <chr>          <dbl>      <dbl>
1 John Smith      NA           2
2 Jane Doe        16          11
3 Mary Johnson    3            1
```

```
pivot_longer(df_messy, -name,
              names_to = "treatment", values_to = "result")
```

# `pivot_longer()` from tidyr package

```
# A tibble: 3 x 3
  name          treatmenta treatmentb
<chr>          <dbl>      <dbl>
1 John Smith    NA          2
2 Jane Doe      16         11
3 Mary Johnson  3           1

pivot_longer(df_messy, -name,
             names_to = "treatment", values_to = "result")
```



# `pivot_longer()` from tidyr package

```
# A tibble: 3 x 3
  name          treatmenta treatmentb
<chr>          <dbl>      <dbl>
1 John Smith    NA          2
2 Jane Doe      16         11
3 Mary Johnson  3          1

pivot_longer(df_messy, -name,
             names_to = "treatment", values_to = "result")
```

# `pivot_longer()` from tidyr package

```
# A tibble: 3 x 3
```

```
  name      treatmenta treatmentb
<chr>      <dbl>      <dbl>
1 John Smith      NA          2
2 Jane Doe        16         11
3 Mary Johnson    3           1
```

```
pivot_longer(df_messy, -name,
              names_to = "treatment", values_to = "result")
```

# `pivot_longer()` from tidyr package

```
# A tibble: 3 x 3
```

```
  name          treatmenta treatmentb
  <chr>          <dbl>      <dbl>
1 John Smith    NA          2
2 Jane Doe     16         11
3 Mary Johnson  3           1
```

```
pivot_longer(df_messy, -name,
              names_to = "treatment", values_to = "result")
```

# `pivot_longer()` result

```
# A tibble: 6 x 3
  name          treatment result
<chr>         <chr>      <dbl>
1 John Smith   treatmenta    NA
2 John Smith   treatmentb     2
3 Jane Doe     treatmenta   16
4 Jane Doe     treatmentb   11
5 Mary Johnson treatmenta     3
6 Mary Johnson treatmentb     1
```

Note: See Appendix for an inverse transformation, `pivot_wider()`

# Try Yourself: “Tidy up” the FANG revenue data

```
# A tibble: 16 x 4
```

	<b>Ticker</b>	<b>Fiscal Year</b>	<b>Quarter</b>	<b>Revenue</b>
	<chr>	<int>	<chr>	<dbl>
1	AMZN	2018	Q1	51.0
2	AMZN	2018	Q2	52.9
3	AMZN	2018	Q3	56.6
4	AMZN	2018	Q4	72.4
5	FB	2018	Q1	12.0
6	FB	2018	Q2	13.2

...

Load the raw “long” data from this URL:

[https://raw.githubusercontent.com/tdmdal/datasets-teaching/main/fang/fang\\_2018.csv](https://raw.githubusercontent.com/tdmdal/datasets-teaching/main/fang/fang_2018.csv)

# Many Ways of Being Messy :(

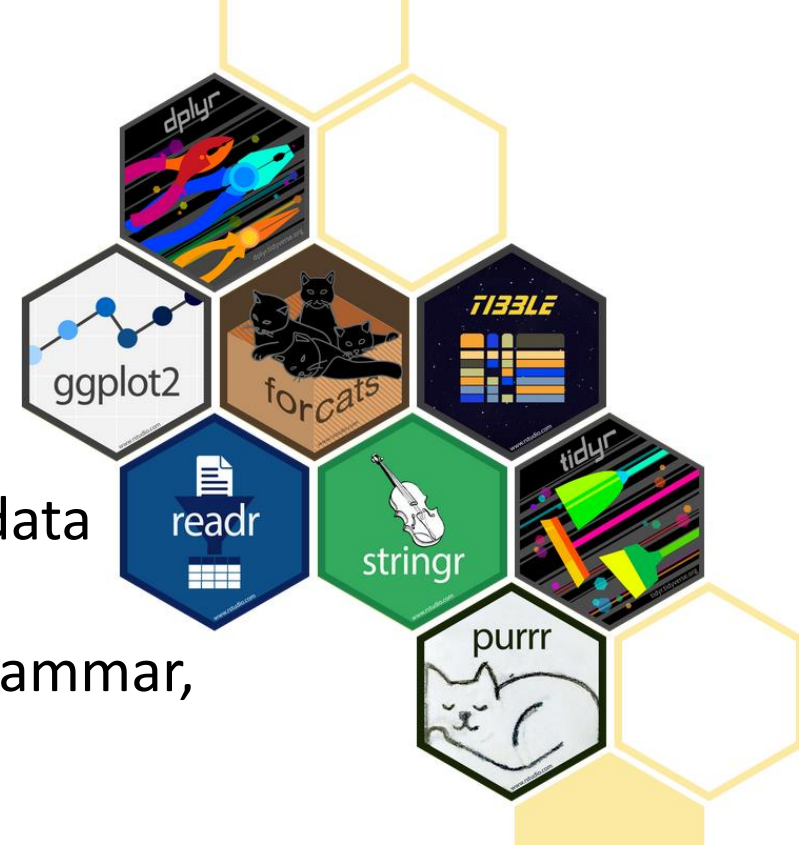
- Messy datasets have 5 common problems (Wickham, 2014)
  1. Column headers are values, not variable names.
  2. Multiple variables are stored in one column.
  3. Variables are stored in both rows and columns.
  4. Multiple types of observational units are stored in the same table.
  5. A single observational unit is stored in multiple tables.

- See another example in Appendix

*“Tidy datasets are all alike, but every messy dataset is messy in its own way.”*

*-- R for Data Science (Chapter 12)*

# Tidy Data and Its Eco-system



- Tidyverse

- “an opinionated collection of R packages designed for data science”
- “All packages share an underlying design philosophy, grammar, and data structures.”

- Other tools in the eco-system (for financial applications)

- tidyquant, a package for quantitative finance
- tidyvert, a set of tidy tools for time series
  - Forecasting: Principles and Practice (3ed), a free book using this toolset



# Plan

- Tidy Data
- Data manipulation (a continuation)
  - summarise() and group\_by()
  - \_join() datasets
- Again, we will focus on basics, underlying principles, and best practices

*“Data Scientists spend up to 80% of the time on data cleaning and 20 percent of their time on actual data analysis”*

-- ??????



# Data manipulation: `dp1yr()`

- Filter observations: `filter()`
- Select variables: `select()`
- Reorder rows: `arrange()`
- Create new variables: `mutate()`
- Collapse column values to a single summary: `summarise()`
  
- Group by: `group by()`

# The Employees Table

```
> employees %>% select(FirstName, LastName, Country)
```

```
# A tibble: 9 x 3
```

```
  FirstName LastName Country
  <chr>      <chr>    <chr>
1 Nancy     Davolio   USA
2 Andrew    Fuller    USA
3 Janet     Leverling USA
4 Margaret  Peacock   USA
5 Steven    Buchanan  UK
```

```
...
```

```
...
```

```
...
```

```
...
```

EmployeeID	LastName	FirstName	Title	...
1	Davolio	Nancy	Sales Representative	...
2	Fuller	Andrew	Vice President, Sales	...
3	Leverling	Janet	Sales Representative	...
4	Peacock	Margaret	Sales Representative	...
...	...	...	...	...

# Count Number of Employees By Country (1)

```
> employees %>% select(FirstName, LastName, Country) %>%  
  group_by(Country)
```

```
# A tibble: 9 x 3
```

```
# Groups:   Country [2]
```

```
  FirstName LastName Country
```

```
  <chr>      <chr>      <chr>
```

```
1 Nancy      Davolio    USA
```

```
2 Andrew    Fuller     USA
```

```
3 Janet     Leverling USA
```

```
...
```

```
...
```

```
...
```

```
...
```

# Count Number of Employees By Country (2)

```
> employees %>% select(FirstName, LastName, Country) %>%  
  group_by(Country) %>%  
  summarise(count = n())
```

```
# A tibble: 2 x 2
```

```
Country count
```

```
<chr>    <int>
```

```
1 UK      4
```

```
2 USA     5
```

# Count Number of Employees By Country (3)

```
> employees %>% select(FirstName, LastName, Country) %>%  
  group_by(Country) %>%  
  summarise(count = n()) %>%  
  arrange(desc(count))
```

```
# A tibble: 2 x 2  
  Country count  
  <chr>    <int>  
1 USA      5  
2 UK       4
```

# More on Data Aggregation

- `summarise()` works with many aggregation functions
  - Aggregation functions: take n inputs, return 1 output
  - e.g. `mean()`, `median()`, `min()`, `max()`, `first()`, `last()`, `n_distinct()`, ...
- There are also Windows functions (useful for time series data)
  - Windows functions: take n inputs, return n output
  - aggregation variations: `cumsum()`, `cummean()`, ...
  - ranking and ordering: `rank()`, `percent_rank()`, ...
  - offsets: `lead()`, `lag()`, ...

<https://dplyr.tidyverse.org/reference/summarise.html>

<https://dplyr.tidyverse.org/articles/window-functions.html>

# Plan

*“Data Scientists spend up to 80% of the time on data cleaning and 20 percent of their time on actual data analysis”*

- Tidy Data

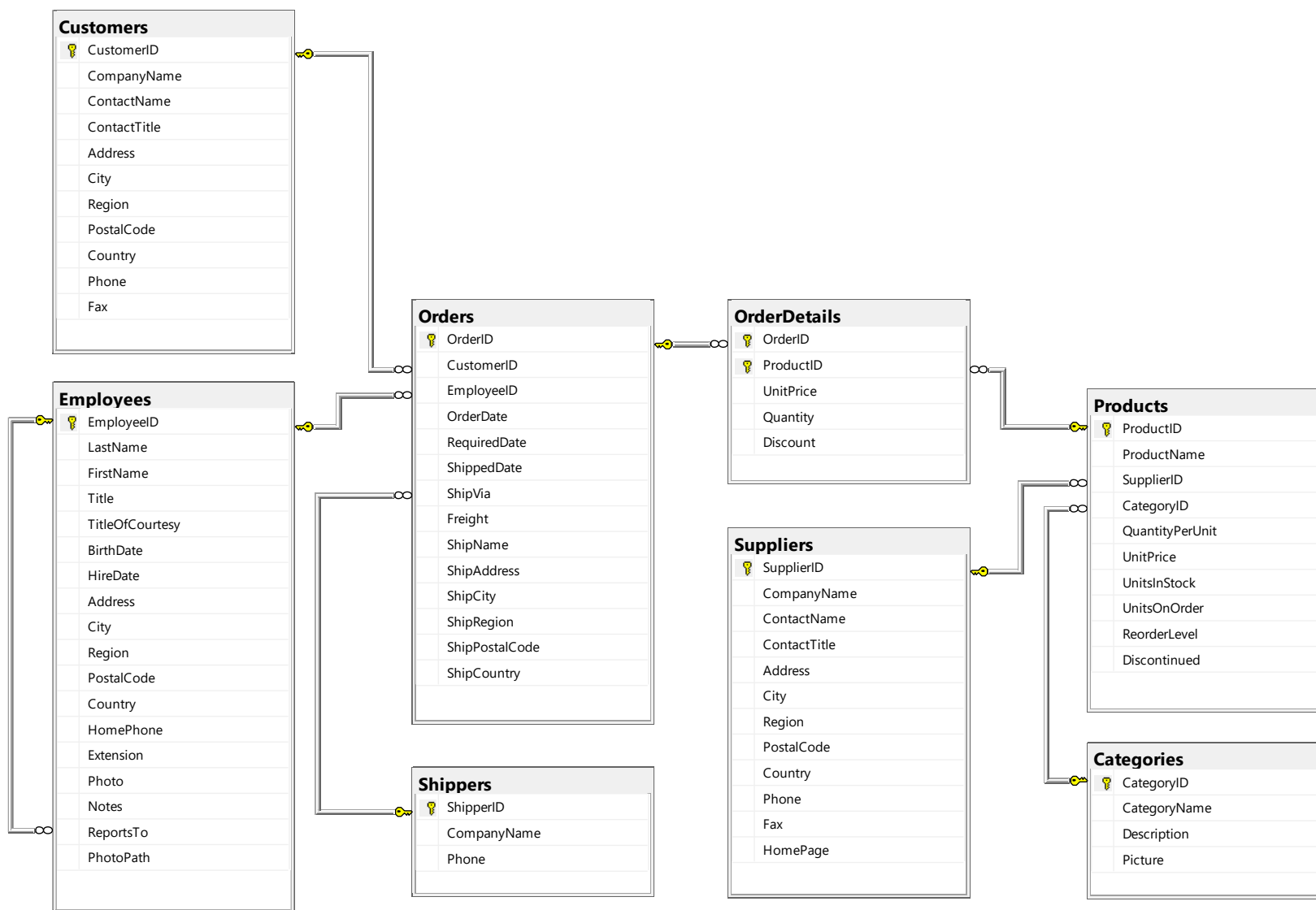
- **Data manipulation** (a continuation)

- `summarise()` and `group_by()`
- **`_join()` datasets**

- Again, we will focus on basics, underlying principles, and best practices

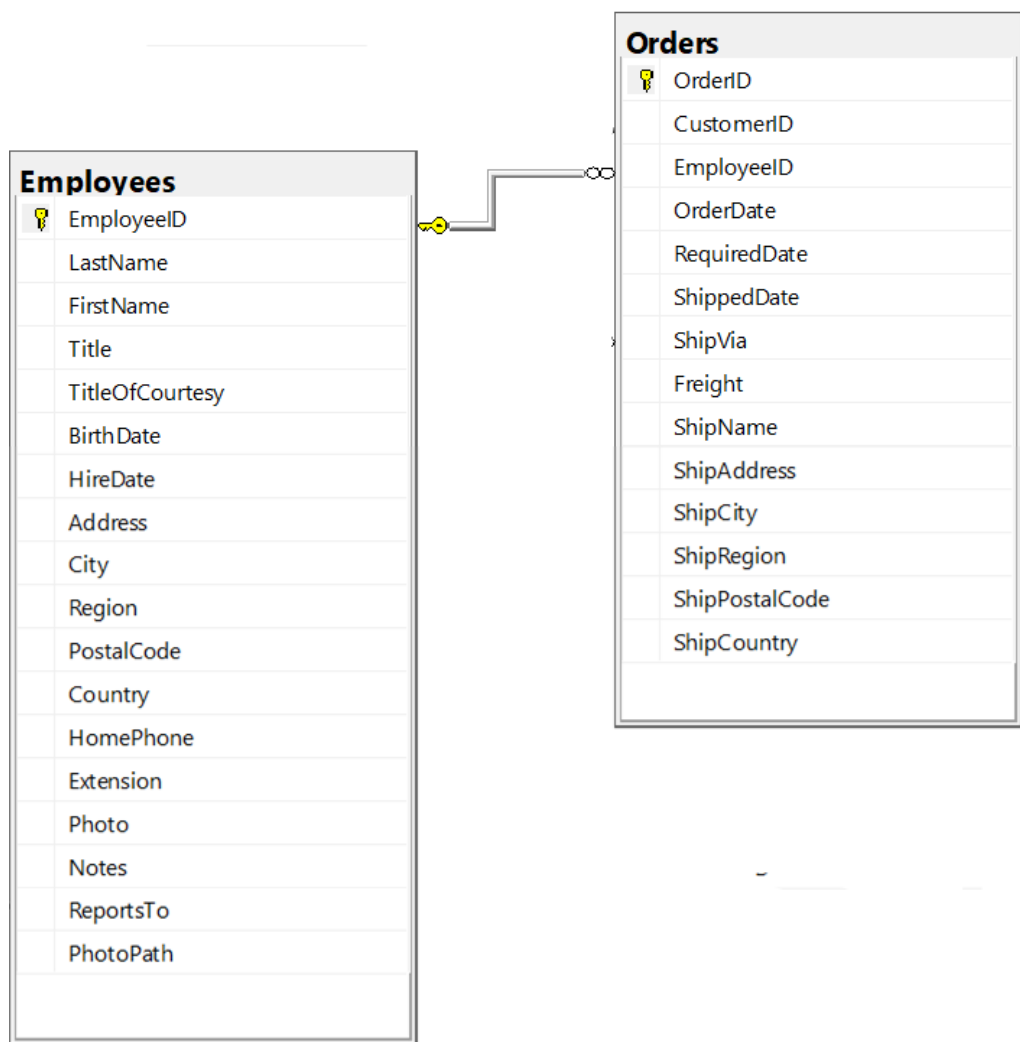
-- ??????

# Motivation: Relation between Datasets/Tables





# Relation between Datasets/Tables – Zoom In



EmployeeID	LastName	FirstName	Title	...
1	Davolio	Nancy	Sales Representative	...
2	Fuller	Andrew	Vice President, Sales	...
3	Leverling	Janet	Sales Representative	...
4	Peacock	Margaret	Sales Representative	...
...	...	...	...	...

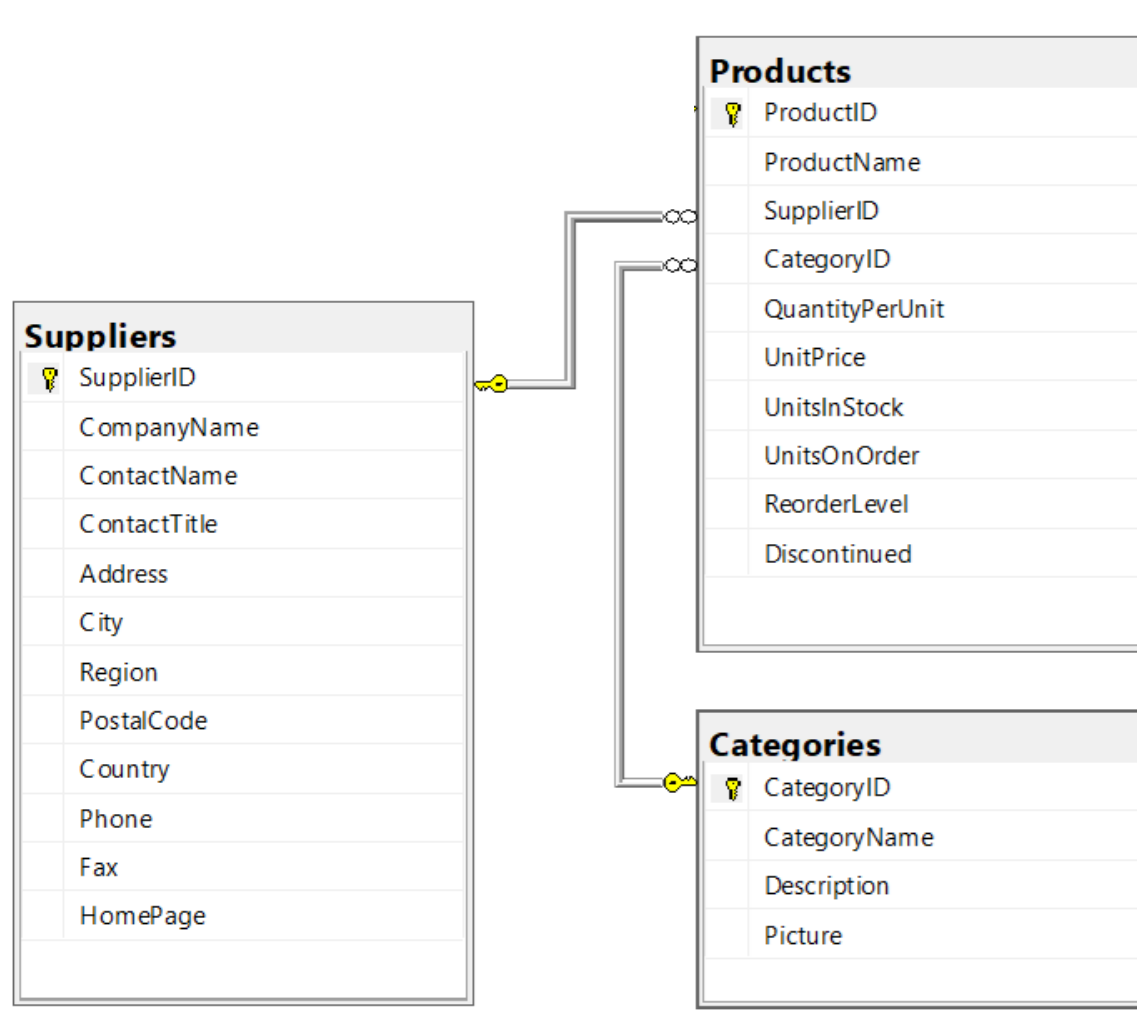
OrderID	CustomerID	EmployeeID	...
10248	VINET	5	...
10249	TOMSP	6	...
10250	HANAR	4	...
...	...	...	...

# Some Terminologies

- Two keys
  - **primary key**: uniquely identifies an observation in its own table
  - **foreign key**: uniquely identifies an observation in another table
- Relationship between tables
  - one-to-one
  - **one-to-many**
  - many-to-many
- Foreign key constraints
  - often imposed on tables in Database, but not on raw data



# Relation between Tables – Another Example



# Join – Inner Join

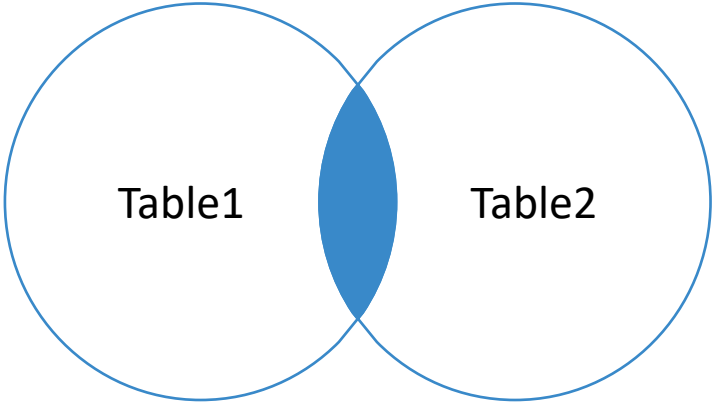


Table1		Table2	
pk	t1c1	fk	t2c1
1	a	1	c
2	b	1	d
		3	e

pk	t1c1	t2c1
1	a	c
1	a	d

```
inner_join(Table1, Table2, by = c("pk" = "fk"))
```

# Join – Left (Outer) Join

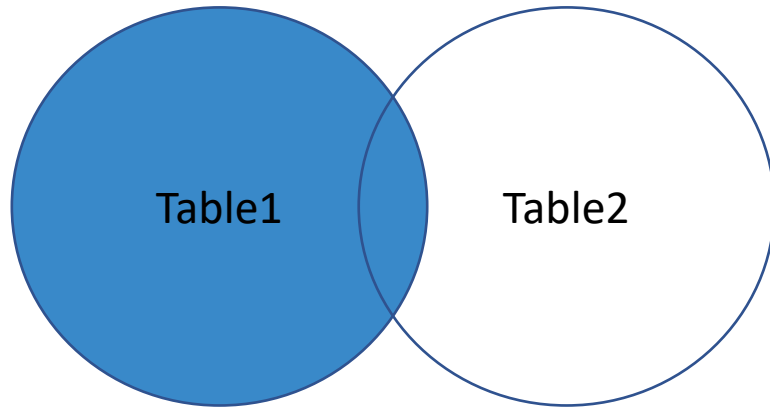


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	t2c1
1	a	c
1	a	d
2	b	NA

```
left_join(Table1, Table2, by = c("pk" = "fk"))
```

# Join - Left (Outer) Join With Exclusion

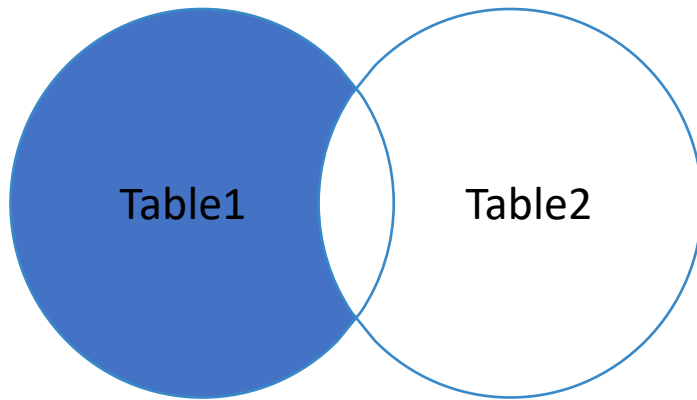


Table1		Table2	
pk	t1c1	fk	t2c1
1	a	1	c
2	b	1	d
		3	e

pk	t1c1	t2c1
2	b	NA

```
Table1 %>%  
  left_join(Table2, by = c("pk" = "fk")) %>%  
  filter(is.na(t2c1))
```

# Appendix

- More on join variations
- `pivot_wide()`, the inverse transformation of `pivot_longer()`
- Another messy data example

# More on Join Variations (learn them yourself)

- More join variation illustrations in the next few slides
  - right join, full join, ...
- Read the “two-table verbs” vignette (in the *dplyr* package doc)
  - <https://dplyr.tidyverse.org/articles/two-table.html>
- Read the reference (see the “two table verbs” section)
  - <https://dplyr.tidyverse.org/reference/index.html>
- For data manipulation tasks in general
  - reading *dplyr* related articles are a good start, <https://dplyr.tidyverse.org/articles/>



# Join – Right (Outer) Join\*

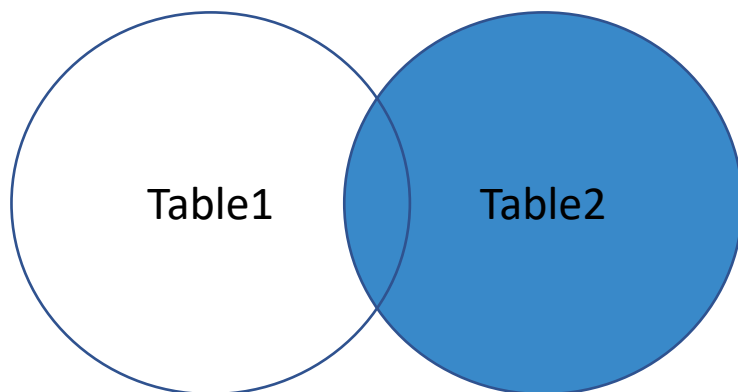


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	t2c1
1	a	c
1	a	d
3	NA	e

```
right_join(Table1, Table2, by = c("pk" = "fk"))
```

Note: can use left\_join as well.

# Join - Right (Outer) Join With Exclusion\*

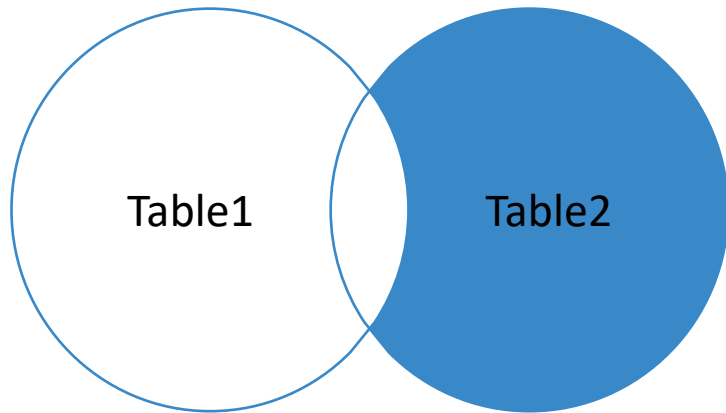


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	t2c1
3	NA	e

```
Table1 %>%
```

```
  right_join(Table2, by = c("pk" = "fk")) %>%
```

```
  filter(is.na(t1c1))
```

# Join – Full Outer Join

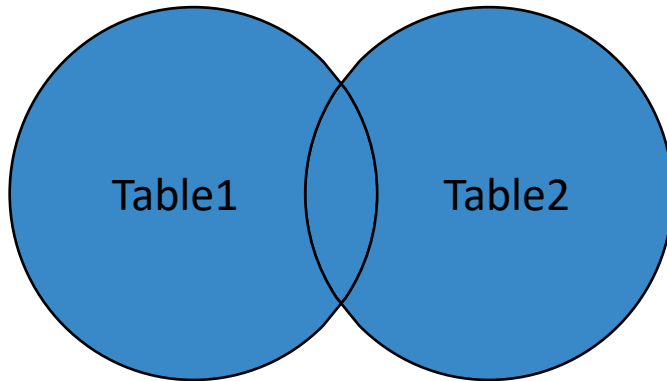


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

```
full_join(Table1, Table2, by = c("pk" = "fk"))
```

pk	t1c1	t2c1
1	a	c
1	a	d
2	b	NA
3	NA	e

# Join – Full Outer Join With Exclusion

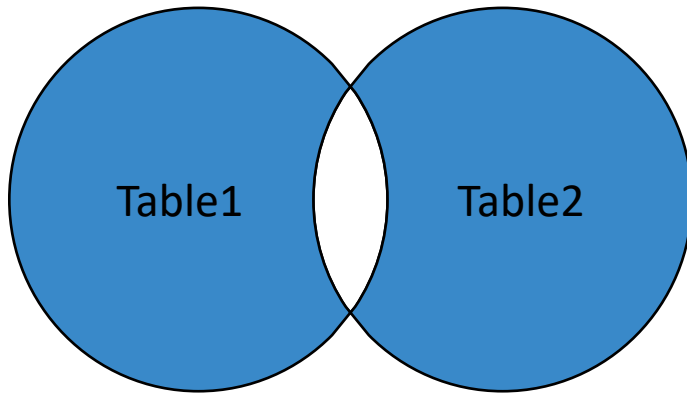


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	t2c1
2	b	NA
3	NA	e

```
Table1 %>%
```

```
  full_join(Table2, by = c("pk" = "fk")) %>%  
  filter(is.na(t1c1) | is.na(t2c1))
```

# The inverse transformation: `pivot_wider()`

	name	treatment	result
	<chr>	<chr>	<dbl>
1	John Smith	a	NA
2	Jane Doe	a	16
3	Mary Johnson	a	3
4	John Smith	b	2
5	Jane Doe	b	11
6	Mary Johnson	b	1

```
pivot_wider(df_tidy,  
              names_from = treatment, values_from = result)
```

<https://tidyr.tidyverse.org/articles/pivot.html>

# The inverse transformation: `pivot_wider()`

	name	treatment	result
	<chr>	<chr>	<dbl>
1	John Smith	a	NA
2	Jane Doe	a	16
3	Mary Johnson	a	3
4	John Smith	b	2
5	Jane Doe	b	11
6	Mary Johnson	b	1

```
pivot_wider(df_tidy,  
            names_from = treatment, values_from = result)
```

# The inverse transformation: `pivot_wider()`

	name	treatment	result
	<chr>	<chr>	<dbl>
1	John Smith	a	NA
2	Jane Doe	a	16
3	Mary Johnson	a	3
4	John Smith	b	2
5	Jane Doe	b	11
6	Mary Johnson	b	1

```
pivot_wider(df_tidy,  
            names_from = treatment, values_from = result)
```

# `pivot_wider()` result

```
# A tibble: 3 x 3
  name          a      b
  <chr>      <dbl> <dbl>
1 John Smith  NA     2
2 Jane Doe   16    11
3 Mary Johnson  3     1
```



# Messy Data – Example 3

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

Table 8: First fifteen rows of the tidied billboard dataset. The `date` column does not appear in the original table, but can be computed from `date.entered` and `week`.

# Messy Data – Example 3

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

Table 8: First fifteen rows of the tidied billboard dataset. The `date` column does not appear in the original table, but can be computed from `date.entered` and `week`.

# The Tidy Version

id	artist	track	time	id	date	rank
1	2 Pac	Baby Don't Cry	4:22	1	2000-02-26	87
2	2Ge+her	The Hardest Part Of ...	3:15	1	2000-03-04	82
3	3 Doors Down	Kryptonite	3:53	1	2000-03-11	72
4	3 Doors Down	Loser	4:24	1	2000-03-18	77
5	504 Boyz	Wobble Wobble	3:35	1	2000-03-25	87
6	98~0	Give Me Just One Nig...	3:24	1	2000-04-01	94
7	A*Teens	Dancing Queen	3:44	1	2000-04-08	99
8	Aaliyah	I Don't Wanna	4:15	2	2000-09-02	91
9	Aaliyah	Try Again	4:03	2	2000-09-09	87
10	Adams, Yolanda	Open My Heart	5:30	2	2000-09-16	92
11	Adkins, Trace	More	3:05	3	2000-04-08	81
12	Aguilera, Christina	Come On Over Baby	3:38	3	2000-04-15	70
13	Aguilera, Christina	I Turn To You	4:00	3	2000-04-22	68
14	Aguilera, Christina	What A Girl Wants	3:18	3	2000-04-29	67
15	Alice Deejay	Better Off Alone	6:50	3	2000-05-06	66

Table 13: Normalised billboard dataset split up into song dataset (left) and rank dataset (right). First 15 rows of each dataset shown; **genre** omitted from song dataset, **week** omitted from rank dataset.

<http://vita.had.co.nz/papers/tidy-data.html>