

Lecture 17: Oct 12, 2018

# Joins

- *Databases*
- *Keys and Relationships*
- *Joins*
  - *Naive, Inner, Full, Left, Right, Anti, Semi*

James Balamuta  
STAT 385 @ UIUC



# Announcements

- **hw06** is due **Friday, Oct 12th, 2018 at 6:00 PM**
- **Office Hour Changes**
  - **John Lee's** are now from **4 - 5 PM** on **WF**
  - **Hassan Kamil's** are now from **2:30 - 3:30 PM** on **TR**
- **Quiz 07** covers Week 6 contents @ [\*\*CBTF\*\*](#).
  - Window: Oct 9th - 11th
  - Sign up: [\*\*https://cbtf.engr.illinois.edu/sched\*\*](https://cbtf.engr.illinois.edu/sched)
- Want to review your homework or quiz grades?  
**Schedule an appointment.**

# Last Time

- **Grammar of Data**
  - Pose question about the data
  - Answer the questions through **five** verbs:  
select, filter, mutate, arrange, and summarise
- **Split-Apply-Combine**
  - **Split** Data into pieces
  - **Apply** function to each piece, and
  - **Combine** result

# Lecture Objectives

- **Explain** the similarities that exist a table in a database and a data frame
- **Apply** the different kinds of join appropriately

# Databases

# Definition:

Database refers to a collection of different tabular pieces of data.

Students				
id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Grades		
student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

Courses	
course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

# Table to Data Frame

... database logic vs R's data structures ...

Students					
Record (Row)	Table (data.frame)				Field (Column)
	id	firstname	lastname	age	instate
	1	Billy	Joe	23	FALSE
	2	Theodore	Squirrel	25	TRUE
	3	Keeya	Nod	21	TRUE

Table Scheme  
(Data Types)

Integer    Character    Character    Integer    Logical

# Why Databases?

... Relational Database Management Systems (RDBMS) ...

- **Speed**
  - High level of optimization around data requests
- **Size**
  - Data in  $R$  is limited by the amount of system memory
- **Scale**
  - Add additional resources to meet computational demand
- **Concurrent**
  - Work on the same data with multiple users without corrupting data

# Databases in R

The screenshot shows a web browser window for the 'Databases using R' website. The URL is <https://db.rstudio.com/>. The page has a navigation bar with links for Overview, Getting Started, Best Practices, Databases, and Blog. On the left, there's a sidebar with sections for Getting Started, Packages, RStudio, and Best Practices, each with a list of related topics. The main content area features a heading 'Databases using R' and a paragraph about RStudio's focus on making it easy to work with databases in R across three key areas. It lists three main sections: 1. RSTUDIO PRODUCTS, 2. USE BEST-IN-CLASS PACKAGES, and 3. PROMOTE BEST PRACTICES. A 'New Connection' dialog box is overlaid on the right side, showing a list of data sources: Spark, AmazonRedshift, Hive, Impala, Oracle, PostgreSQL, Salesforce, and SQLServer. The 'SQLServer' option is selected. A 'Using RStudio Connections' link and a 'Cancel' button are also visible in the dialog.

## Databases using R

At RStudio, we are working to make it as easy as possible to work with databases in R. This work focuses on **three key areas**:

### 1. RSTUDIO PRODUCTS

- The new RStudio [Connections Pane](#) makes it possible to easily connect to a variety of data sources, and **explore the objects and data** inside the connection
- To RStudio commercial customers, we offer [RStudio Professional ODBC Drivers](#), these are data connectors that help you connect to some of the most popular databases.

### 2. USE BEST-IN-CLASS PACKAGES

Build and/or document how to use packages such as: [dplyr](#), [DBI](#), [odbc](#), [keyring](#) and [pool](#)

### 3. PROMOTE BEST PRACTICES

This website is the main channel to provide support in this area. RStudio is also working through other delivery channels, such as upcoming webinars and in-person training during our RStudio conferences.

[Read more →](#)

## Latest Announcements

<https://db.rstudio.com/>

# Keys and Relationships

## Definition:

*Primary Key* refers to a unique set of values in one or more columns that is used to identify the rows of a table.

Students

<b>id</b>	<b>firstname</b>	<b>lastname</b>	<b>age</b>	<b>instate</b>
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Primary Key →



[Source](#)

# Definition:

*Foreign Keys* refer to a set of one or more columns that is a primary key in another table.

The diagram illustrates the relationship between two tables: **Students** and **Grades**.

**Students Table:**

Students				
<b>Primary Key</b>	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeyaa	Nod	21	TRUE

**Grades Table:**

Grades		
<b>Foreign Key</b>	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	CS374	A
3	HIST101	C-

States		Capitals		
state	id	state_id	capital_id	capital
Illinois	1	1	1	Springfield
California	2	2	2	Sacramento
Texas	3	3	3	Austin

Mother		Children		
Mother	id	mother_id	child_id	child_name
Lily	1	1	1	James
Maggie	2	1	2	Susie
		2	3	Aj

Doctors		DoctorsToPatients		Patients	
dr	id	dr_id	patient_id	patient_id	p_name
Brown	1	1	1	1	Arman
Patel	2	2	1	2	Qihui
		2	2		

# Types of Relationships

- **One-to-one:** one row in a table matches exactly with only one row in another table
- 

- **One-to-many (Many-to-one):** one row of a table matches *multiple rows* in another table.
- 
- 

- **Many-to-many:** *multiple rows* in one table can be mapped to *multiple rows* in another table and vice versa.
- 
- 

# Your Turn

1. Identify the different keys for each database table.
2. What kinds of relationships exist between tables?

Students

<b>id</b>	<b>firstname</b>	<b>lastname</b>	<b>age</b>	<b>instate</b>
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Grades

<b>student_id</b>	<b>course_id</b>	<b>grade</b>
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

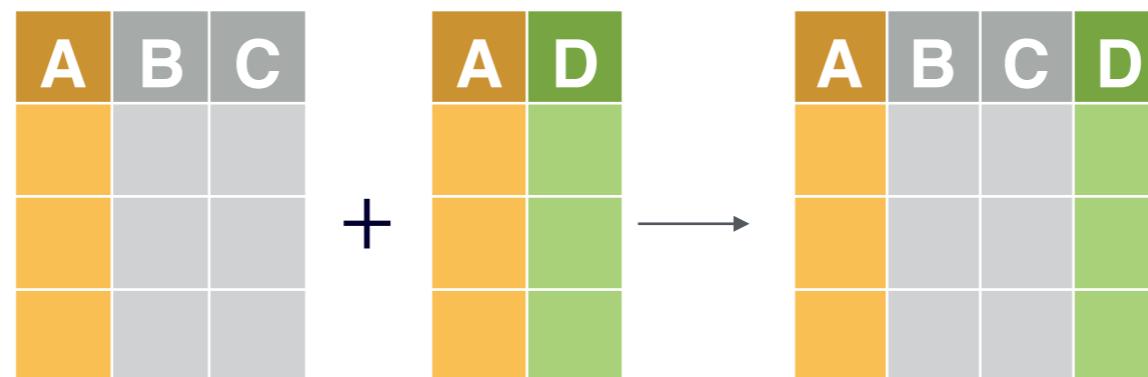
Courses

<b>course_id</b>	<b>acronym</b>
STAT385	SPM
STAT432	BSL
HIST100	GH

# Joins

## **Definition:**

*Joining* or *merging* refers to combining two different pieces of data together to form a larger data set that contains more observations, variables, or both.



# Ordered Naive Joins

... merging data naively ...

```
# Same number of rows, exact ordering, no repeated columns.  
first_df = data.frame(A = c(1, 2, 3, 4),  
                      B = c("A", "B", "C", "A"))  
sec_df = data.frame(D = c(38.4, 39.9, 40, 20.5))  
  
# Merge the data together  
merged_df = data.frame(first_df, sec_df)  
# Or, bind by column  
merged_df_cols = cbind(first_df, sec_df)  
  
# Retrieve specific columns with the same order  
selected_df = data.frame(first_df$A, sec_df$D)
```

# Ordering for Naive Joins

... when data isn't ordered right ...

```
# Same number of rows, exact ordering, no repeated columns.
```

```
bad_first_df = data.frame(A = c(4, 3, 2, 1),  
                           B = c("A", "C", "B", "A"))
```

```
bad_sec_df = data.frame(A = c(2, 1, 4, 3),  
                           D = c(39.9, 38.4, 20.5, 40))
```

```
# Order data frames
```

```
ordered_first_df = bad_first_df[order(bad_first_df$A), ]
```

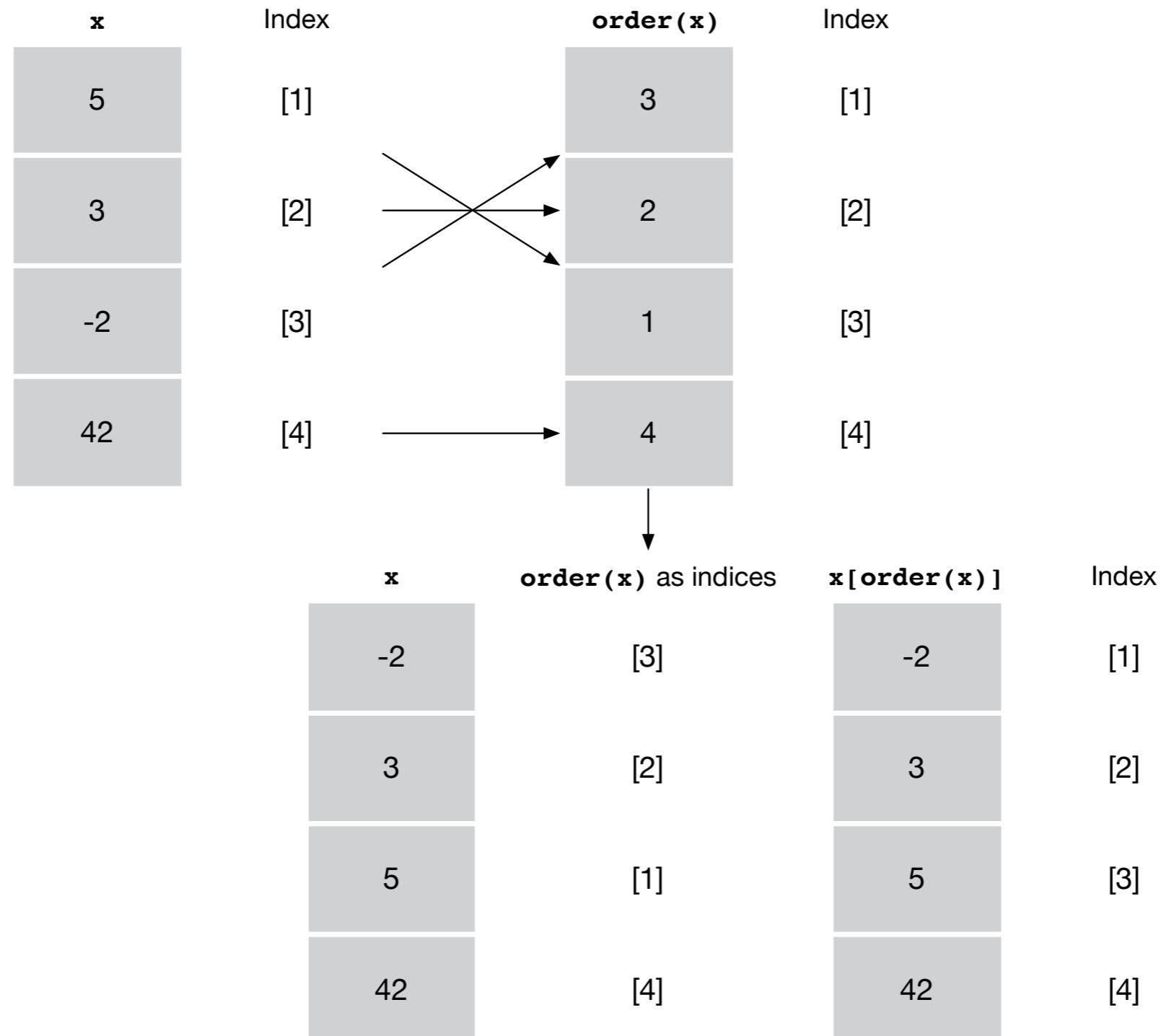
```
ordered_sec_df = bad_sec_df[order(bad_sec_df$A), ]
```

```
# Combine the ordered data frames
```

```
ordered_merged_df = data.frame(ordered_first_df$A,  
                               ordered_first_df$B,  
                               ordered_sec_df$D)
```

# Behind `order()`

... using sorted positional indices we can rearrange the data ...



# Naive Joining **Fails with Uneven Rows**

# Types of Joins

A diagram illustrating a mutating join. On the left, there are two tables represented as grids. The first table has columns labeled A, B, and C, with the first column colored orange and the others grey. The second table has columns labeled A and D, with the first column orange and the second green. A plus sign (+) is placed between them, followed by a right-pointing arrow. To the right of the arrow is a third table with four columns labeled A, B, C, and D. The first three columns (A, B, C) are orange, and the fourth column (D) is green.

A	B	C	
orange			
orange			
orange			

+

A	D
orange	
orange	
orange	

→

A	B	C	D
orange			
orange			
orange			

A diagram illustrating a filtering join. On the left, there are two tables represented as grids. The first table has columns labeled A, B, and C, with all columns blue. The second table has columns labeled A and D, with the first column blue and the second pink. A plus sign (+) is placed between them, followed by a right-pointing arrow. To the right of the arrow is a third table with four columns labeled A, B, C, and D, all colored blue.

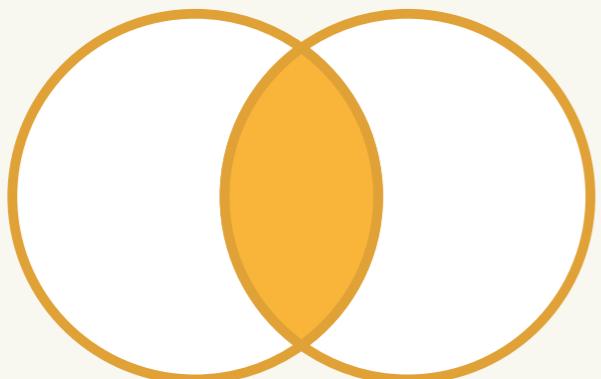
A	B	C	
blue			
blue			
blue			

+

A	D
blue	
blue	
blue	

→

A	B	C	D
blue			
blue			
blue			



- **Mutating joins** will add new variables to one table from matching observations in another.
- **Filtering joins** will filter observations from one table based on whether or not they match an observation in the other table.
- **Set operations** will treat observations as if they were elements in a set.

# Joins for Uneven Data

... how to handle different numbers of observations ...

`inner_join(x, y)`

1	x1	y1
2	x2	y2

**Original**

**X**      **y**

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

Source

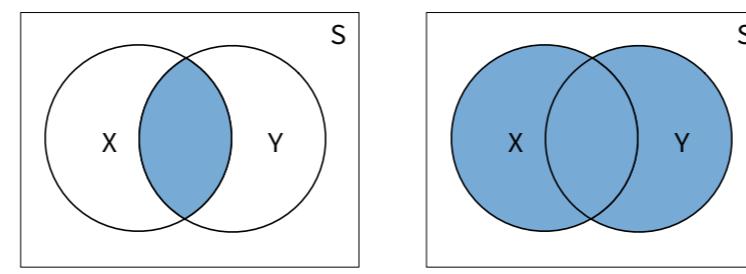
`full_join(x, y)`

1	x1	y1
2	x2	y2
3	x3	
4		y4
<u>Source</u>		

`left_join(x, y)`

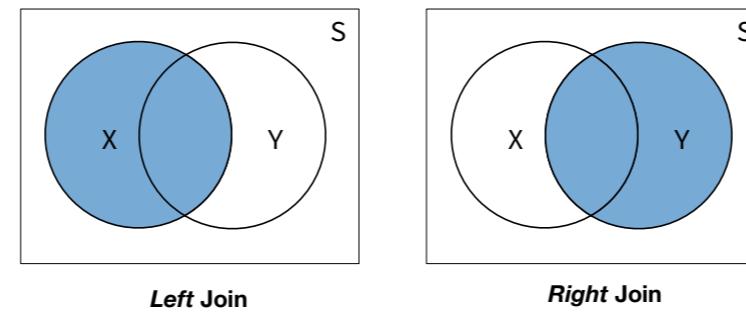
1	x1	y1
2	x2	y2
3	x3	

JOIN Venn Diagrams



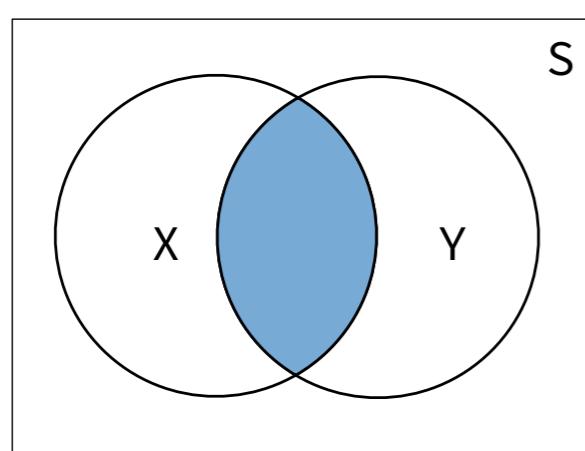
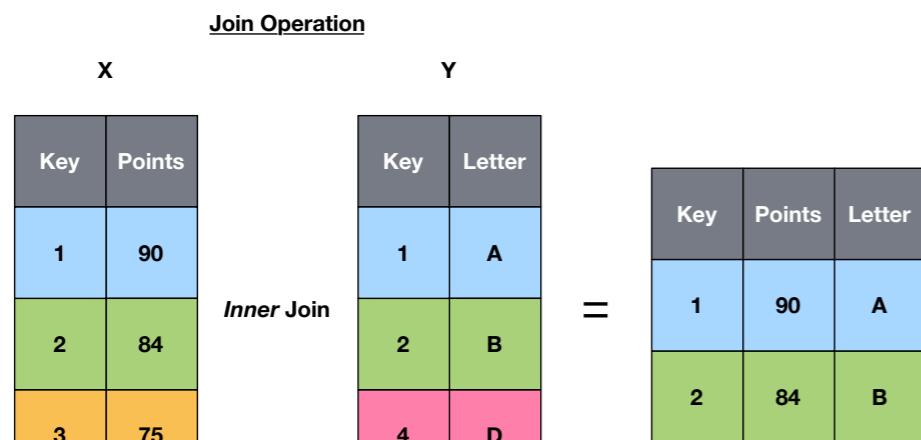
`right_join(x, y)`

1	x1	y1
2	x2	y2
4		y4



# Inner Join

acquires the set of values that are in both **Table X** and **Table Y**.



```
# Using inner_join in dplyr  
dplyr::inner_join(X, Y, by = "Key")
```

```
# Using Base R's merge()  
# function to perform an inner  
# join  
merge(X, Y, by = "Key")
```

# Full (Outer) Join

acquires the set of all values in **Table X** and **Table Y**, regardless of whether they have values that exist in both tables. If the values do not exist, the missing side will have **NA** values substituted.

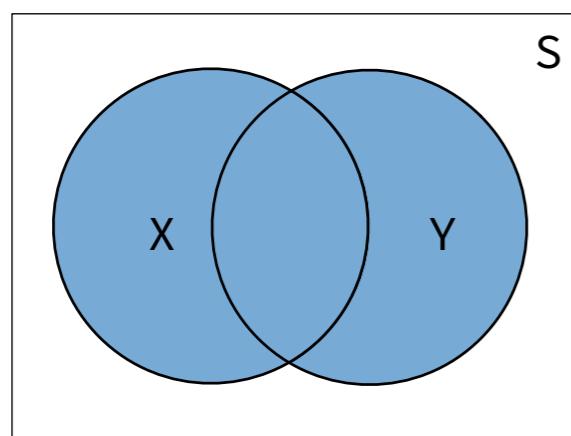
Join Operation

X		Y			
Key	Points	Key	Letter		
1	90	1	A		
2	84	2	B		
3	75	4	D		

*Full Join*

=

Key	Points	Letter
1	90	A
2	84	B
3	75	NA
4	NA	D



```
# Using full_join in dplyr  
dplyr::full_join(X, Y, by = "Key")
```

```
# Using Base R's merge()  
# function to perform a full  
# join  
merge(X, Y, by = "Key",  
      all.x = TRUE,  
      all.y = TRUE)
```

# Left (Outer) Join

acquires the set of complete values in **Table X** paired with the values in **Table Y** if available. If the values do not exist, the left side will have **NA** values substituted.

Join Operation

**X**

Key	Points
1	90
2	84
3	75

**Y**

Key	Letter
1	A
2	B
4	D

**Left Join**

=

Key	Points	Letter
1	90	A
2	84	B
3	75	NA

Venn Diagram

```
# Using left_join in dplyr  
dplyr::left_join(X, Y, by = "Key")
```

```
# Using Base R's merge()  
# function to perform a left  
# join  
merge(X, Y, by = "Key",  
      all.x = TRUE)
```

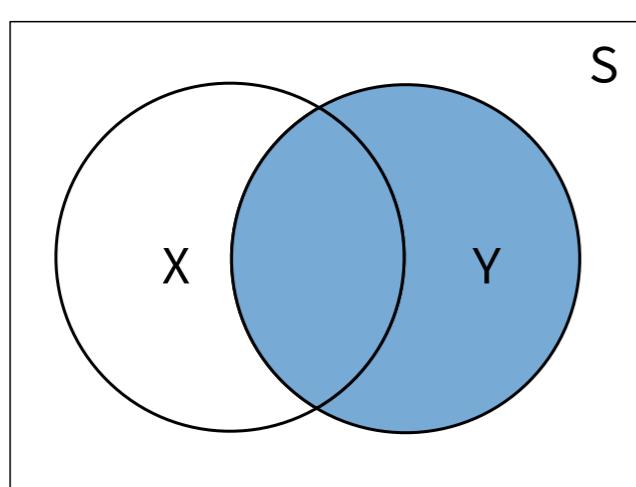
# Right (Outer) Join

acquires the set of complete values in **Table Y** paired with the values in **Table X** if available. If the values do not exist, the right side will have **NA** values substituted.

Join Operation

X		Y		=		
Key	Points	Key	Letter	Key	Points	Letter
1	90	1	A	1	90	A
2	84	2	B	2	84	B
3	75	4	D	4	NA	D

*Right Join*



```
# Using right_join in dplyr  
dplyr::right_join(X, Y, by = "Key")
```

```
# Using Base R's merge()  
# function to perform a left  
# join  
merge(X, Y, by = "Key",  
      all.y = TRUE)
```

# Your Turn

Join together the different tables in the **student database**.

Students				
id	firstname	lastname	age	instate
1	Billy	Joe	23	FALSE
2	Theodore	Squirrel	25	TRUE
3	Keeya	Nod	21	TRUE

Grades		
student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

Courses	
course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

Would the following joins be equivalent? If so, why?

```
dplyr::left_join(X, Y, by = "Key")  
dplyr::right_join(Y, X, by = "Key")
```

# Filtering Joins

... match vs. no match ...

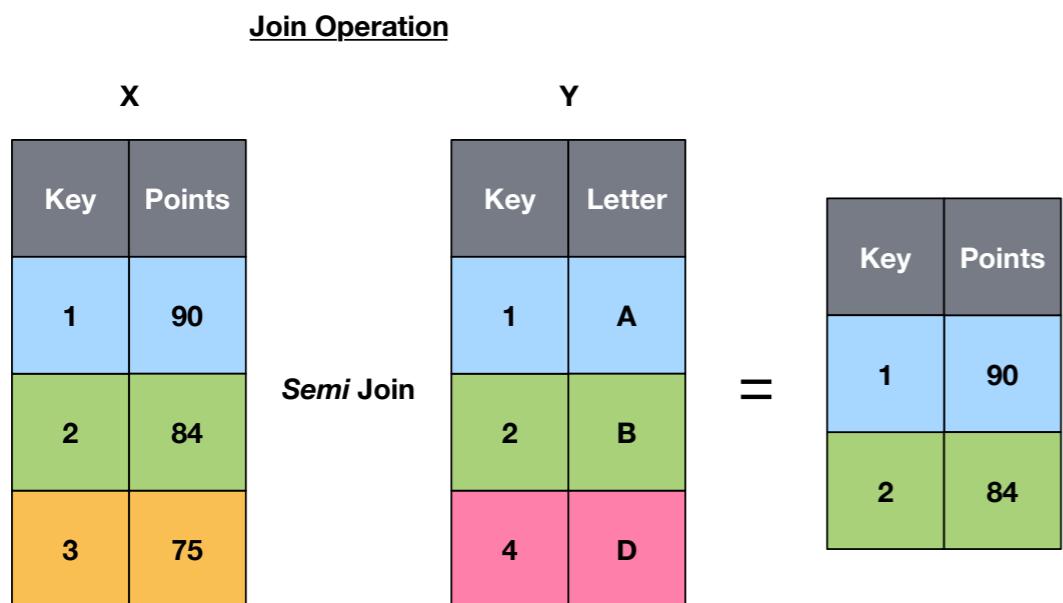
**Original**

	x	y
semi_join(x, y)	1 2 3	x1 x2 x3
	1 2	y1 y2 y4
anti_join(x, y)		3 x3

[Source](#)

# Semi joins

acquires the set of complete values in **Table X** that have a matching key in **Table Y**.



```
# Using semi_join in dplyr  
dplyr::semi_join(X, Y, by = "Key")
```

# Anti joins

purges the set of complete values in **Table X** that have a matching key in **Table Y**.

```
# Using anti_join in dplyr  
dplyr::anti_join(X, Y, by = "Key")
```

Join Operation

X		Y	
Key	Points	Key	Letter
1	90	1	A
2	84	2	B
3	75	4	D

*Anti Join*

=

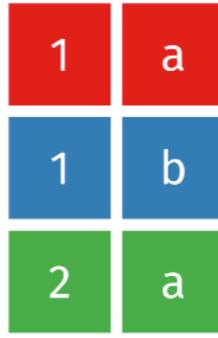
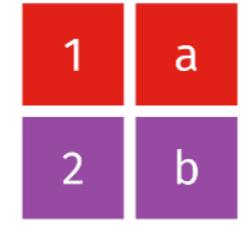
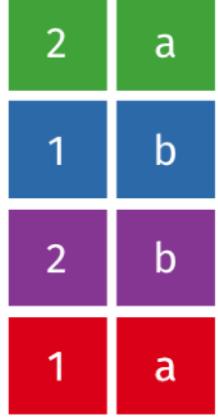
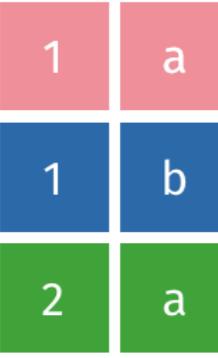
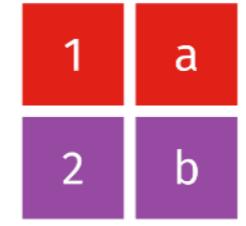
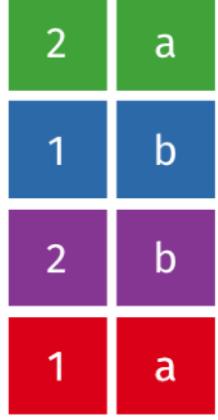
Key	Points
3	75

# Your Turn

1. Install the **fueleconomy** package.
2. Determine the appropriate keys between **common** and **vehicles** tables.
3. Perform a semi join

# Set Manipulations

... operating on data ...

		Original		
		x	y	
<code>setdiff(x, y)</code>				
<code>union(y, x)</code>				
<code>setdiff(y, x)</code>				
<code>intersect(x, y)</code>				

[Source](#)

# Set Operations

```
x = c(-8, 0, 2, 1, 23, NA)  
y = c(-8, 3, 1, NA, 2, 10)
```

```
union(x, y)      # X or Y (Full)  
# [1] -8 0 2 1 23 NA 3 10
```

```
intersect(x, y)  # X and Y (Intersect)  
# [1] -8 2 1 NA
```

```
setdiff(x, y)    # Y - X (Anti-join)  
# [1] 0 23
```

```
setdiff(y, x)    # X - Y (Anti-join)  
# [1] 3 10
```

```
setequal(x, y)   # X = Y  
# [1] FALSE
```

```
is.element(x, y) # X in Y (Intersect)  
# [1] TRUE FALSE TRUE TRUE FALSE TRUE
```

```
x %in% y       # equivalent  
# [1] TRUE FALSE TRUE TRUE FALSE TRUE
```

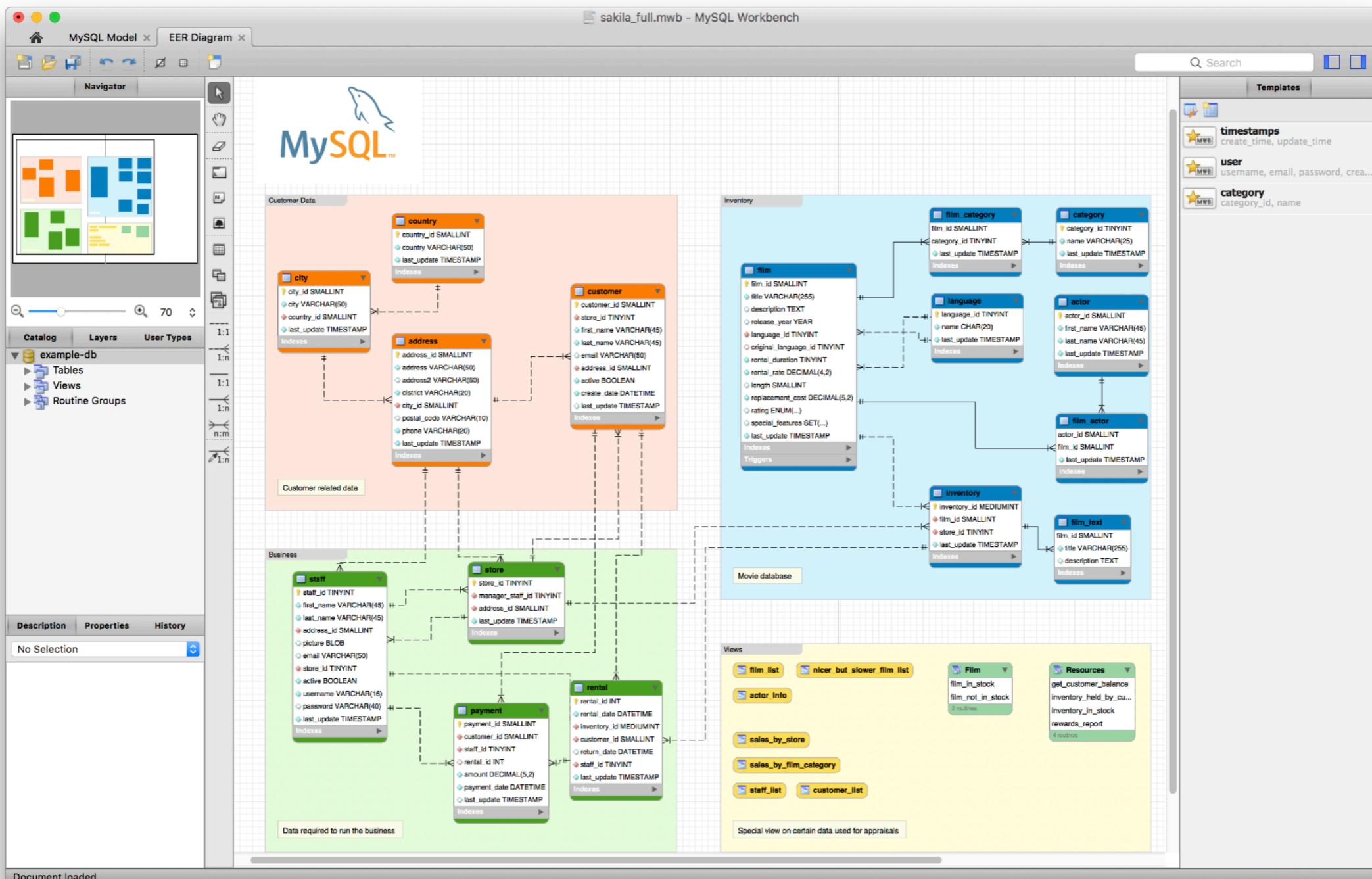
# Recap

- **Databases**
  - Collection of data
  - Data tables mirror *R*'s data frame
- **Keys and Relationships**
  - Keys are unique field(s) to identify rows in data.
  - Relationships show how data is connected between tables
- **Joins**
  - Naively merging data is rarely a good idea.
  - Mutating, Filtering, and Set Joins are better for a varying number of rows between data sets.

# Resources

# MySQL Workbench

... GUI for Designing Databases ...



<https://www.mysql.com/products/workbench/>

This work is licensed under the  
Creative Commons  
Attribution-NonCommercial-  
ShareAlike 4.0 International  
License

