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
- 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

Table
(data.frame)

Field (Column)

	id	firstname	lastname	age	instate
Record (Row)	1	Billy	Joe	23	FALSE
	2	Theodore	Squirrel	25	TRUE
	3	Keeya	Nod	21	TRUE
	Integer	Character	Character	Integer	Logical

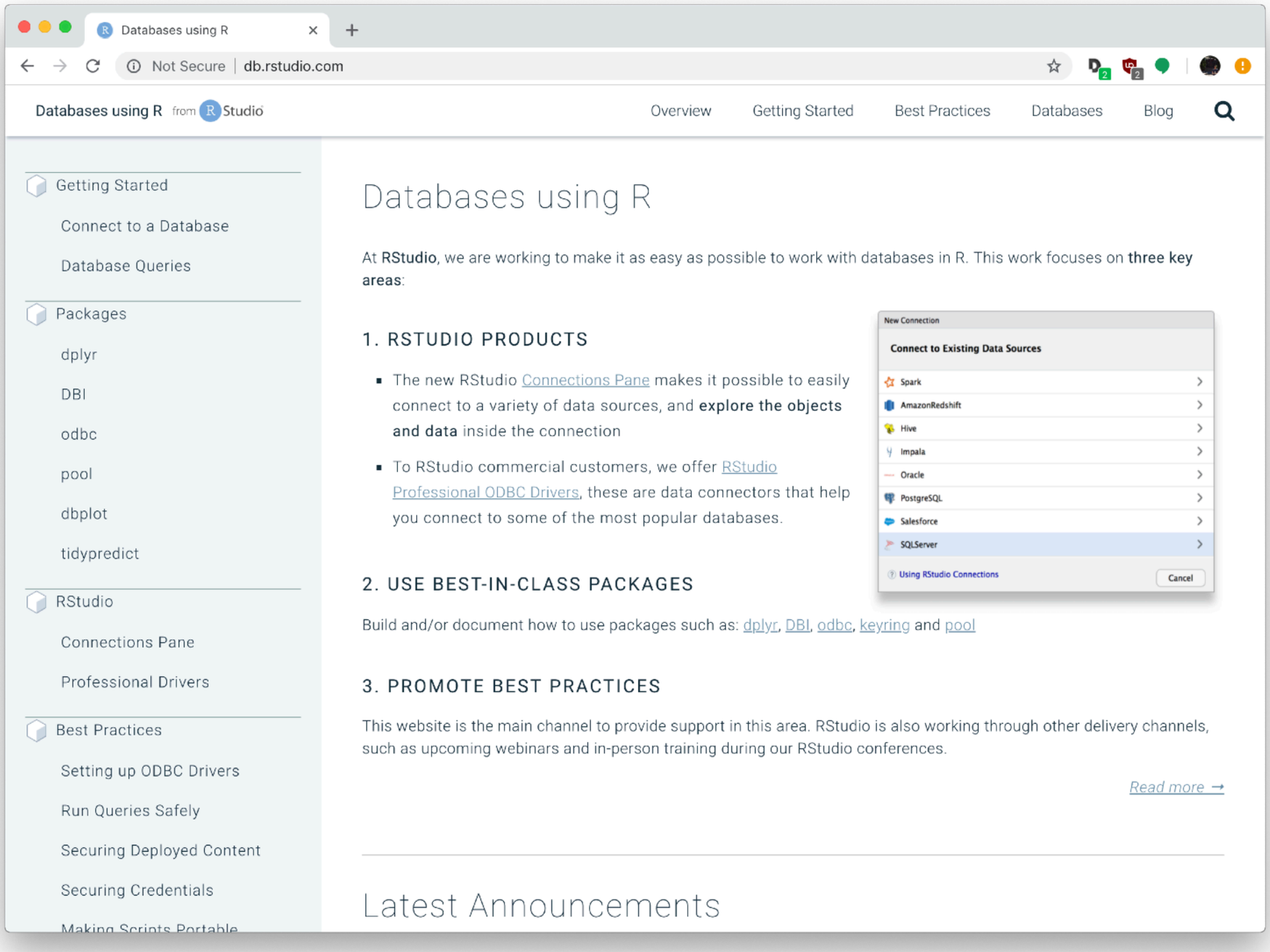
Table Scheme
(Data Types)

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



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.

Primary Key →

Students

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

Source

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

Students

Primary Key

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

Grades

Foreign Key

student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	CS374	A
3	HIST101	C-

States

state	id
Illinois	1
California	2
Texas	3

Capitals

state_id	capital_id	capital
1	1	Springfield
2	2	Sacramento
3	3	Austin

Mother

Mother	id
Lily	1
Maggie	2

Children

mother_id	child_id	child_name
1	1	James
1	2	Susie
2	3	Aj

Doctors

dr	id
Brown	1
Patel	2

DoctorsToPatients

dr_id	patient_id
1	1
2	1
2	2

Patients

patient_id	p_name
1	Arman
2	Qihui

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 a 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

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

Grades

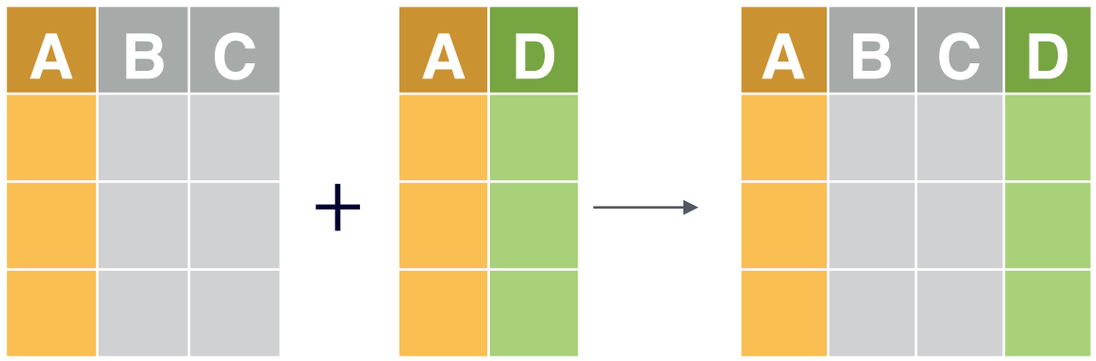
student_id	course_id	grade
1	STAT385	A+
2	STAT432	A-
1	HIST100	A
3	STAT385	B+

Courses

course_id	acronym
STAT385	SPM
STAT432	BSL
HIST100	GH

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 …

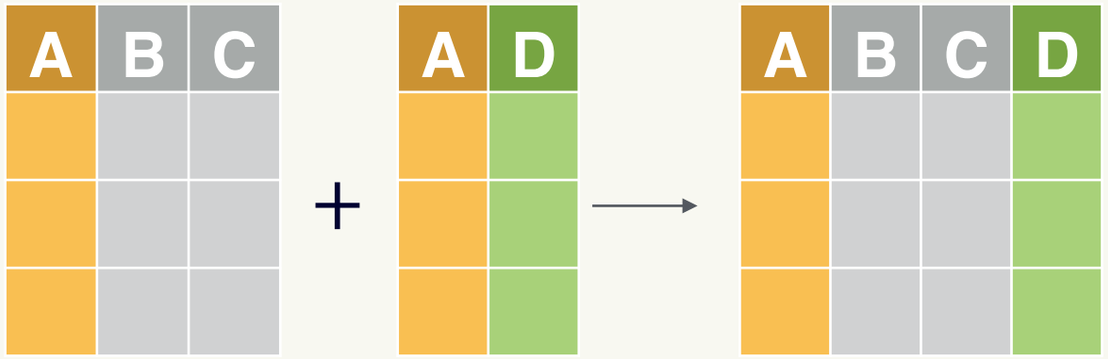
`x`	Index
5	[1]
3	[2]
-2	[3]
42	[4]

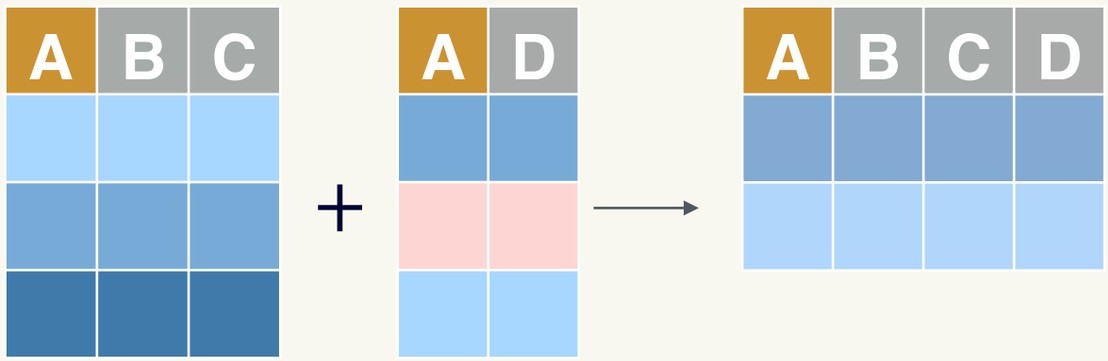
`order(x)`	Index
3	[1]
2	[2]
1	[3]
4	[4]

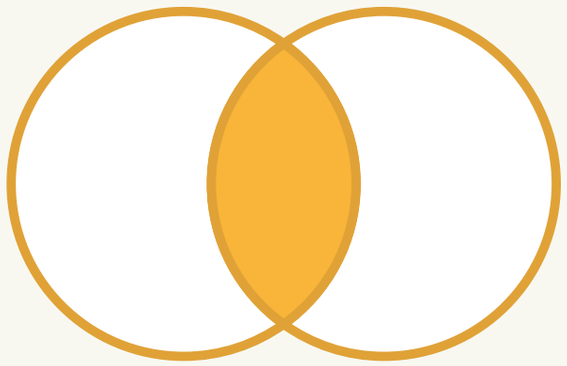
`x`	`order(x)` as indices	`x[order(x)]`	Index
-2	[3]	-2	[1]
3	[2]	3	[2]
5	[1]	5	[3]
42	[4]	42	[4]

Naive Joining

Fails with Uneven Rows







Types of Joins

- **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

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y4

Source

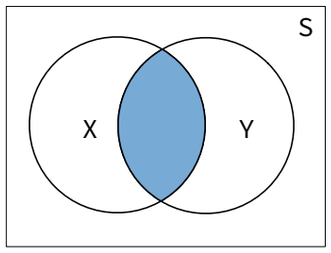
`full_join(x, y)`

1	x1	y1
2	x2	y2
3	x3	
4		y4

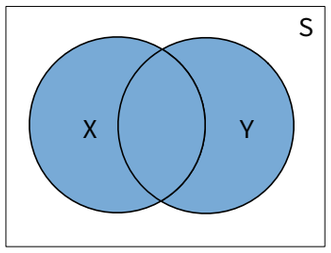
`left_join(x, y)`

1	x1	y1
2	x2	y2
3	x3	

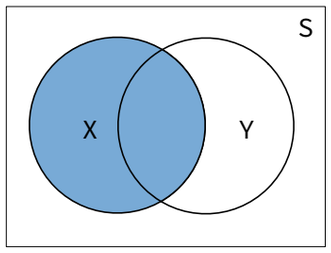
JOIN Venn Diagrams



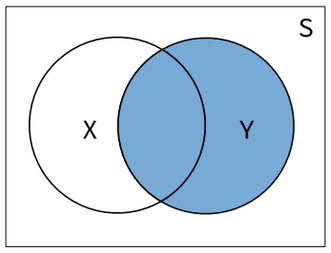
***Inner* Join**



***Full* Join**



***Left* Join**



***Right* Join**

`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***.

Join Operation

X **Y**

Key	Points
1	90
2	84
3	75

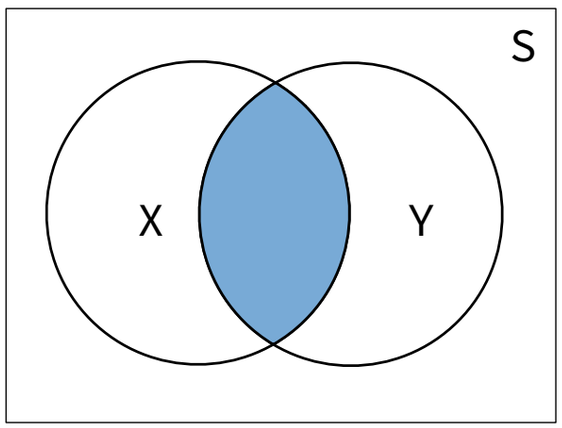
***Inner* Join**

Key	Letter
1	A
2	B
4	D

=

Key	Points	Letter
1	90	A
2	84	B

Venn Diagram




```
# 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

Key	Points
1	90
2	84
3	75

Full **Join**

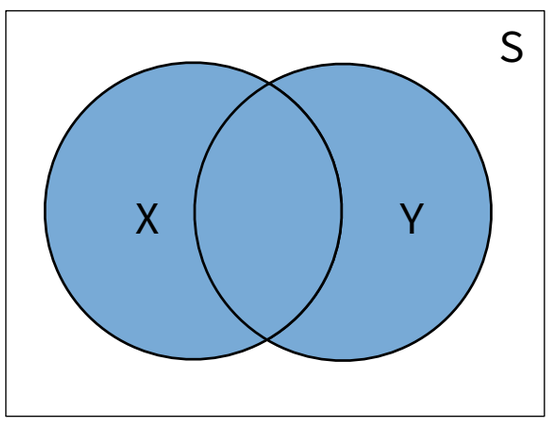
Y

Key	Letter
1	A
2	B
4	D

=

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

Venn Diagram




```
# 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

***Left* Join**

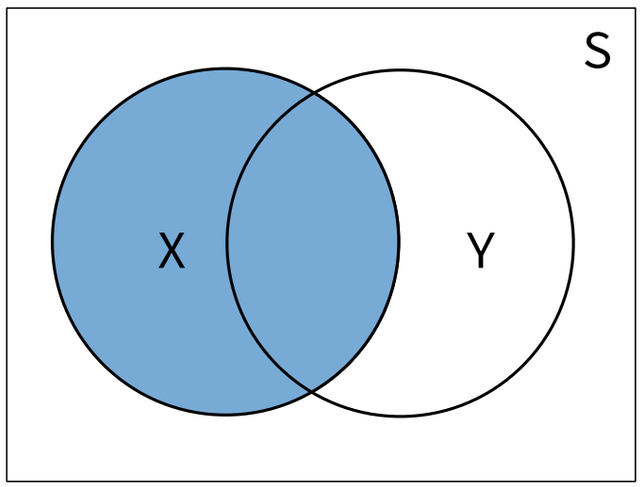
Y

Key	Letter
1	A
2	B
4	D

=

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

Key	Points
1	90
2	84
3	75

***Right* Join**

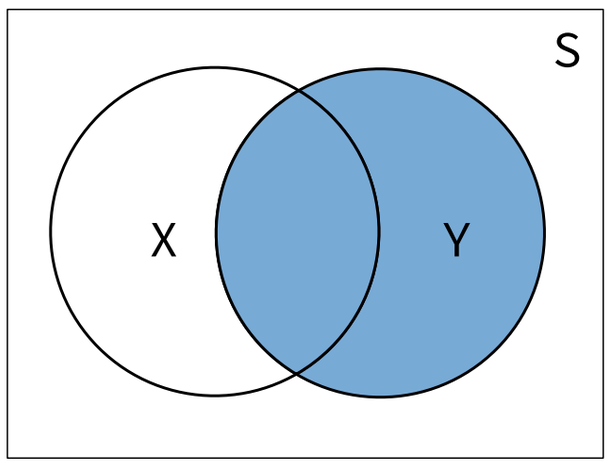
Y

Key	Letter
1	A
2	B
4	D

=

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

Venn Diagram




```
# 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

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y4

`semi_join(x, y)`

1	x1
2	x2

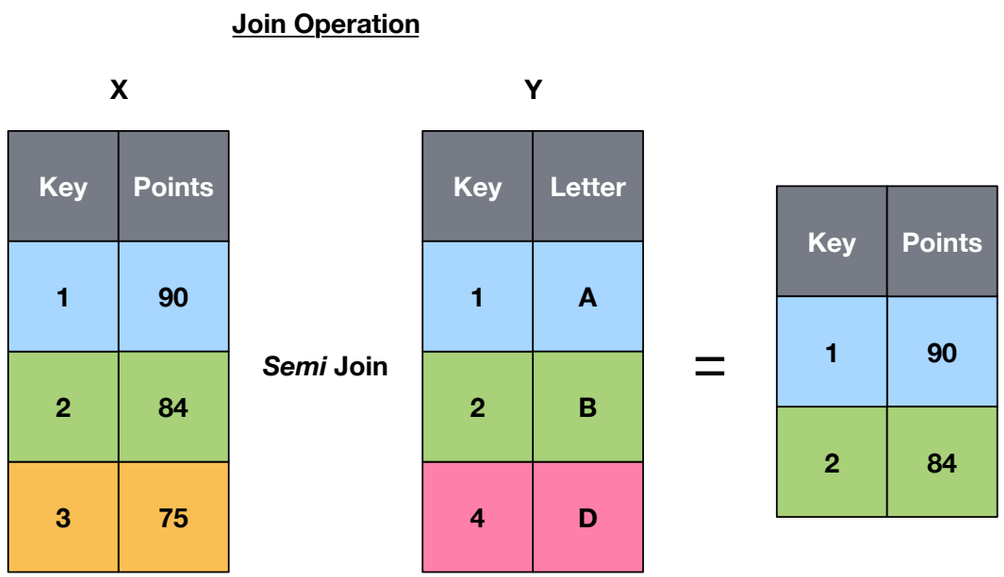
`anti_join(x, y)`

3	x3

Source

Semi joins

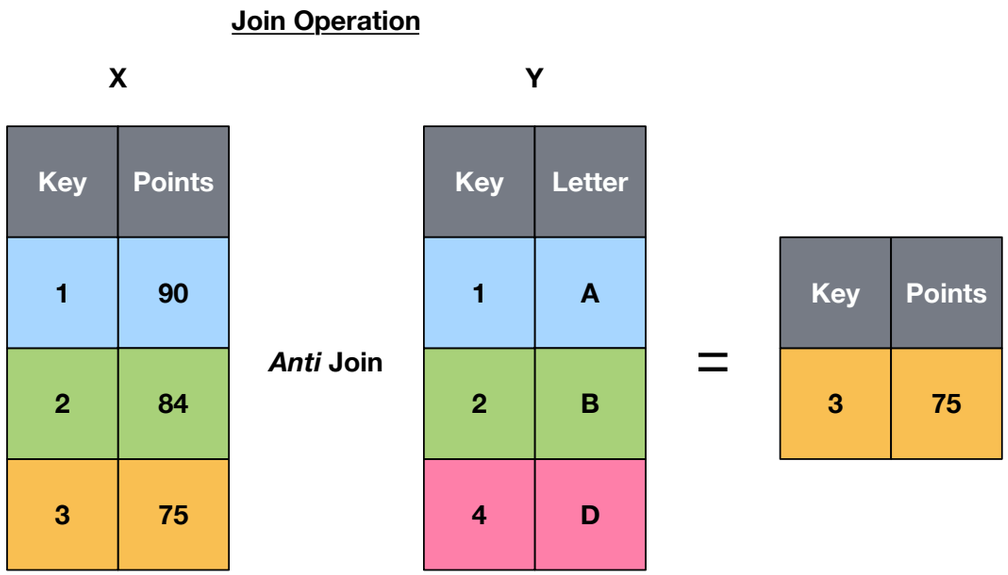
acquires the set of complete values in ***Table X*** that have a matching key in ***Table Y***.




```
# 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")
```

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

1	a
1	b
2	a

y

1	a
2	b

`setdiff(x, y)`

1	b
2	a

`setdiff(y, x)`

2	b

`union(y, x)`

2	a
1	b
2	b
1	a

`intersect(x, y)`

1	a

Source

Set Operations

```r
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
```

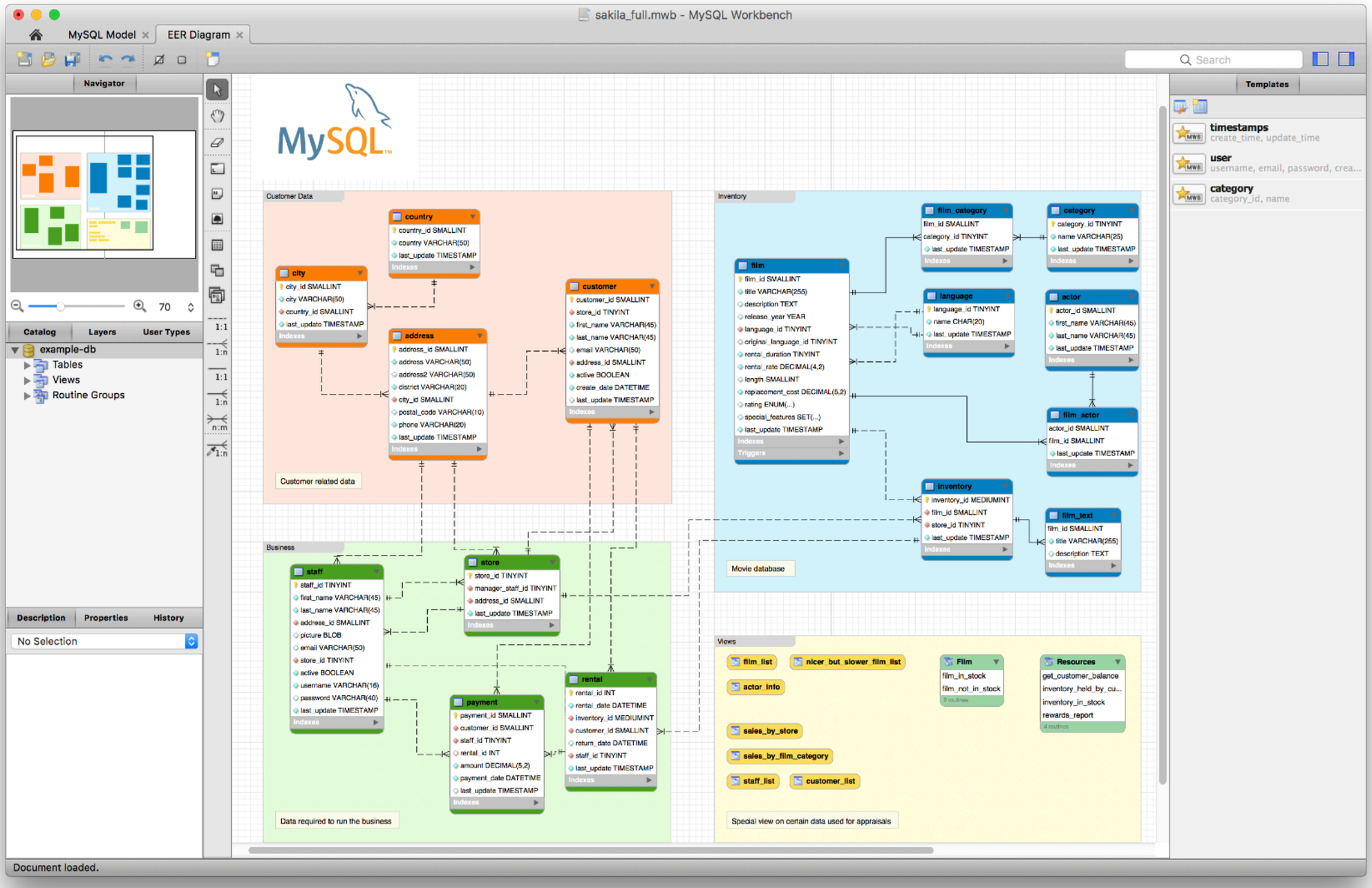
Recap

- **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/