

# Import



# DBI

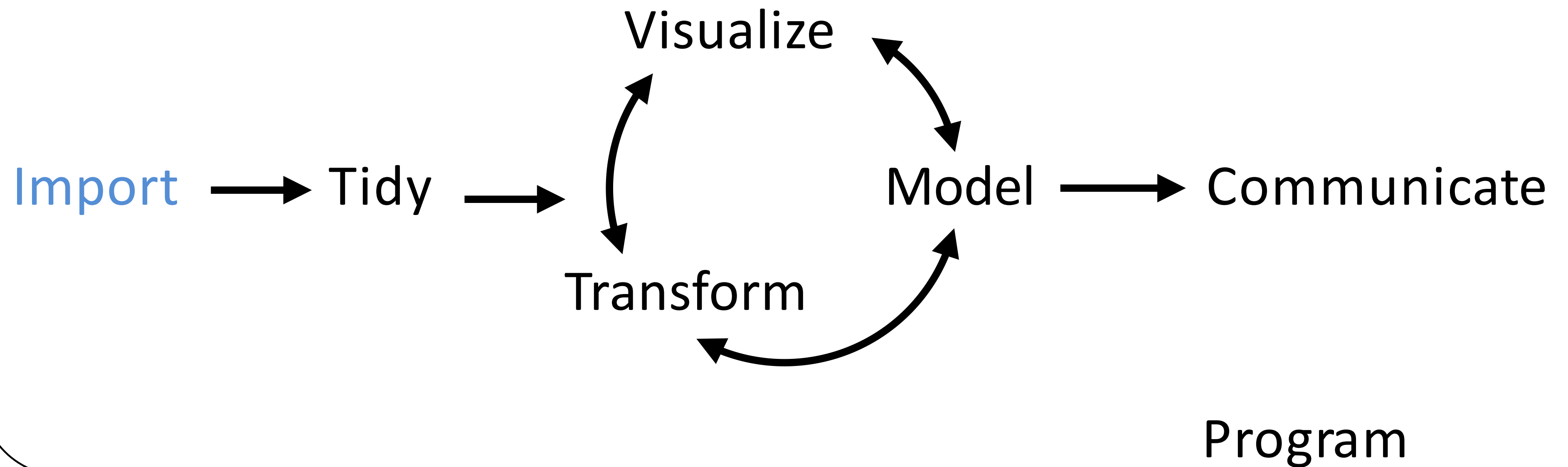


“I ~~rob banks~~ use databases  
because its where the ~~money~~  
data is.”

—Willie Sutton



# (Applied) Data Science



# Connecting to SQL Server



# Connecting to SQL

1. Click Database
2. Click Driver Manager
3. Click New
4. Put the following settings:
  - a) Driver Type: Generic
  - b) Driver Name: SQL Server
  - c) Class Name: `com.microsoft.sqlserver.jdbc.SQLServerDriver`
  - d) URL Template: `jdbc:sqlserver://{host}[:{port}][;databaseName={database}]`
  - e) Default Port: 1433
  - f) Default Database: master
5. Move to the Libraries tab

# Connecting to SQL

1. On the Libraries Tab
2. Click Add File
  - a) Navigate to C:/Program Files/Dbeaver/plugins
  - b) Open mssql-jdbc-12.4.1.jre11.jar
3. Click Add File again
  - a) Navigate to C:/Program Files/Dbeaver/jre/bin
  - b) Change the file type selector to \*.\*
  - c) Open mssql-jdbc\_auth-12.4.1.x64.dll
4. Click Ok
5. Click Database
6. Click New Database Connection

# Connecting to SQL

1. Select your Generic SQL Server from the All tab
2. Enter the following properties
  - a) Host: DEVSQL17\CountyStat\_DW
  - b) Port: 50900
  - c) Database/Schema: HealthCare\_DataWarehouse
  - d) Username: Your T#
  - e) Password: Your Windows Password
3. Click Ok
4. Double Click the new connection from the Database Navigator

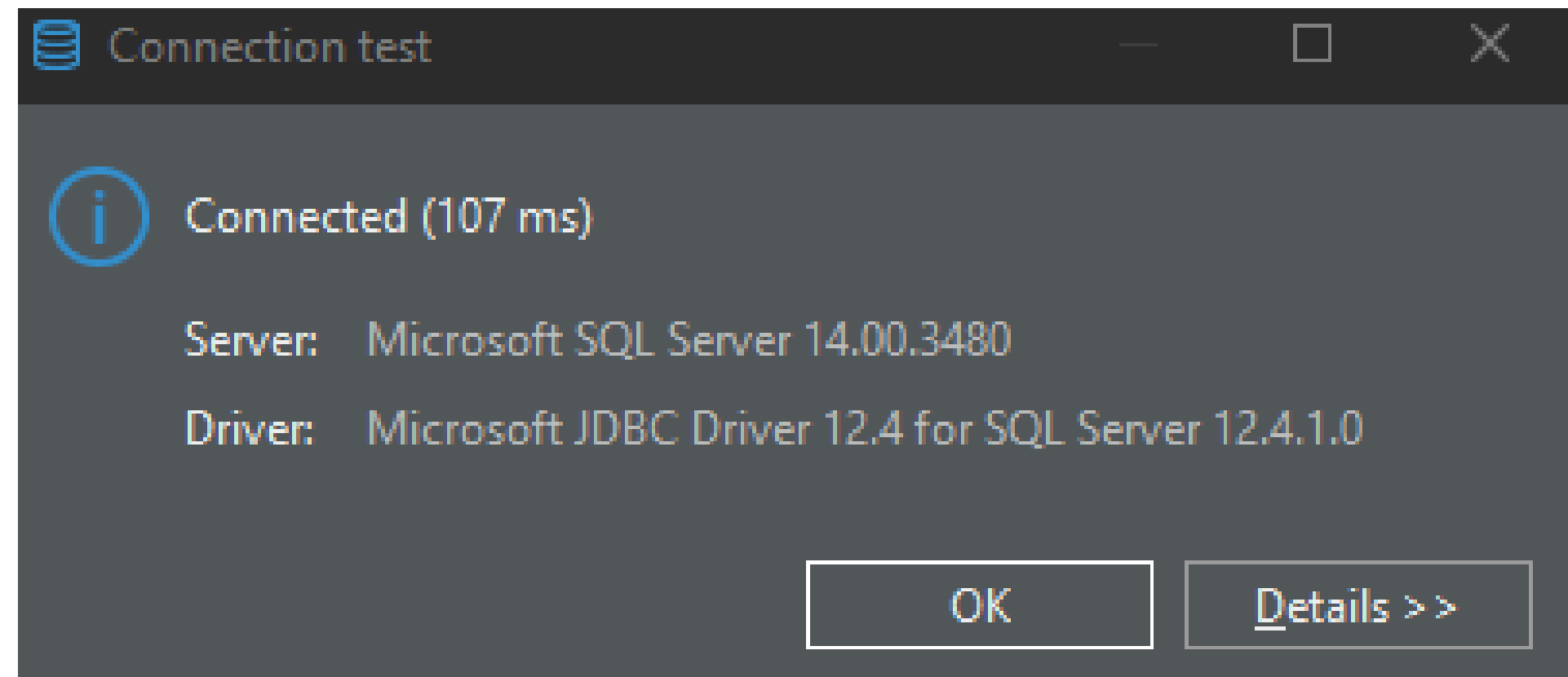
# Connecting to SQL

1. Select your Generic SQL Server from the All tab
2. Enter the following properties
  - a) Host: DEVSQL17\CountyStat\_DW
  - b) Port: 50900
  - c) Database/Schema: HealthCare\_DataWarehouse
  - d) Username: Your T#
  - e) Password: Your Windows Password
3. Click Ok
4. Double Click the new connection from the Database Navigator
  - a) The connection should fail
5. Right click the Connection and select Edit Connection



# Connecting to SQL

1. Navigate to the Driver properties Tab
  - a) Verify the encrypt property is set to true
  - b) Set the integratedSecurity property to true
  - c) Set the trustServerCertificate property to true
2. Click Test Connection
  - a) If you get this screen click OK
  - b) If not raise your hand



# Writing SQL



# SQL IDE'S

- ▶ There are a bunch of SQL IDE's each database provider has their own
- ▶ If you're in a workplace like mine with no standard then I suggest something like DBeaver because it connects to pretty much everything
- ▶ If not, then use whatever comes standard with the platform



# SQL





“SQL is a domain specific  
language used in programming  
and ... data held in a relational  
database management  
system”

—Wikipedia



# Structuring a Query





# QUERIES

ORDER	CLAUSE	FUNCTION
1	from	Choose and join tables to get base data.
2	where	Filters the base data.
3	group by	Aggregates the base data.
4	having	Filters the aggregated data.
5	select	Returns the final data.
6	order by	Sorts the final data.
7	limit	Limits the returned data to a row count.

Source: [periscope data](#)

# EXERCISE



- ▶ Open Dbeaver
- ▶ Highlight the Dev Warehouse Connection
- ▶ Create a new SQL Script
  - ▶ SQL > New SQL Script (or CTRL+J)
- ▶ Build a query that selects Vaccine Site locations from the Health\_SalesForce\_Site table

A small, colorful, abstract graphic with a grid-like pattern in shades of yellow, orange, and blue. The text "2:00" is overlaid in white.

**2:00**

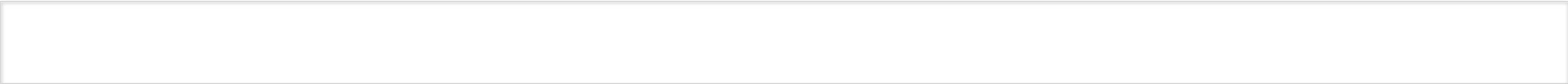


# SOLUTION

```
SELECT * FROM ACHD.Health_SalesForce_Site
```

DBeaver Short Cut:

- Right click the table you want to start a query from in the Database Navigator
- Click Read data in SQL console
  - Note: This method names all of the columns instead of using \*





# WHERE



# BETWEEN ... AND

- ▶ BETWEEN

- ▶ *Grab Values between two other values, like IN but for numeric values*
- ▶ *Works like < and >*

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1
AND value2;
```

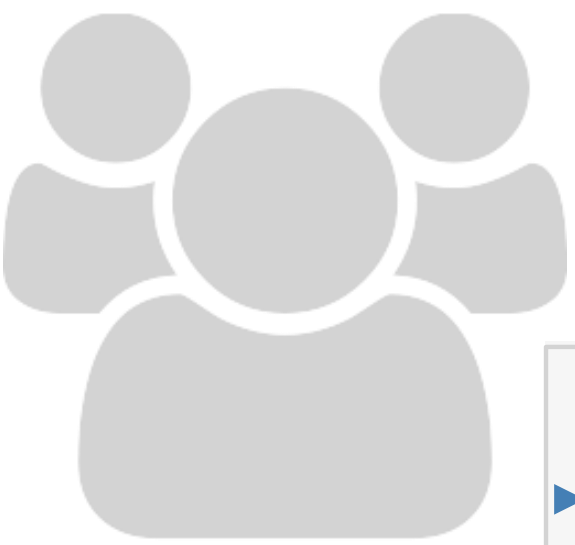
# IN STATEMENTS

- ▶ Useful for when you have an input that returns multiple
- ▶ This works the same way %in% does in R
- ▶ Checks to see if the value in the column matches *any* of the values in your list

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name IN (value1, value2, ...)
```



# EXERCISE



- ▶ This time let's target Health\_SalesForce\_Appointment
- ▶ Use the BETWEEN function as a WHERE filter to get Second Dose appointments from April 1<sup>st</sup> to the 30<sup>th</sup> in 2021.
- ▶ Stretch goal: Use the IN Filter to get requests where the status is either "Day Of Screening Question Incomplete" or "Cancelled".

A square icon with a colorful, abstract, pixelated background. Overlaid on this background is the text "5:00" in a bold, white, sans-serif font.

**5:00**



# SOLUTION

```
SELECT * FROM ACHD.Health_SalesForce_Appointment
WHERE Dose_Number__c = 'Second Dose' AND
      Appointment_Date_Time__c BETWEEN '2021-04-01' AND '2021-04-30'
```

```
SELECT * FROM ACHD.Health_SalesForce_Appointment
WHERE Dose_Number__c = 'Second Dose' AND
      Appointment_Date_Time__c BETWEEN '2021-04-01' AND '2021-04-30' AND
      Status__c IN ('Cancelled', 'Day Of Screening Question Incomplete')
```

# Pop Quiz

Tidyverse equivalent?

`filter()`

# SELECT Functions and GROUP BY



# SQL FUNCTIONS

- ▶ Sometimes you don't just want the raw data
- ▶ You want to aggregate the data in SQL before you load it into R
  - ▶ Use another server to do the heavy lifting so you don't have to!



# DISTINCT

- ▶ DISTINCT()
  - ▶ Every unique value of a column.
  - ▶ Placing TWO columns inside will return unique instances of both columns:

```
DISTINCT("REQUEST_TYPE", "DEPARTMENT")
```

# Pop Quiz

Tidyverse equivalent?

`distinct()`

# MATH FUNCTIONS

- ▶ **MIN()**
  - ▶ Returns minimum value in a column(s)
- ▶ **MAX()**
  - ▶ Return max value in a column(s)

# Pop Quiz

Tidyverse equivalent?

`min(), max()`

# COUNT, AVERAGE, SUM

- ▶ COUNT() - returns the number of rows that your query returns
  - ▶ SELECT COUNT(column\_name)  
FROM table\_name
- ▶ AVG() - returns the average value of a numeric column.
  - ▶ SELECT AVG(column\_name)  
FROM table\_name
- ▶ SUM() - function returns the total sum - numeric columns only
  - ▶ SELECT SUM(column\_name)  
FROM table\_name



# Pop Quiz

R/Tidyverse equivalents?

`n()`, `mean()`, `sum()`

# GROUP BY

- ▶ This is helpful for when you are doing any of the summary functions mentioned in the previous slides. (COUNT, SUM, MAX etc)
- ▶ Any column that isn't handled with a function should be included in your GROUP BY

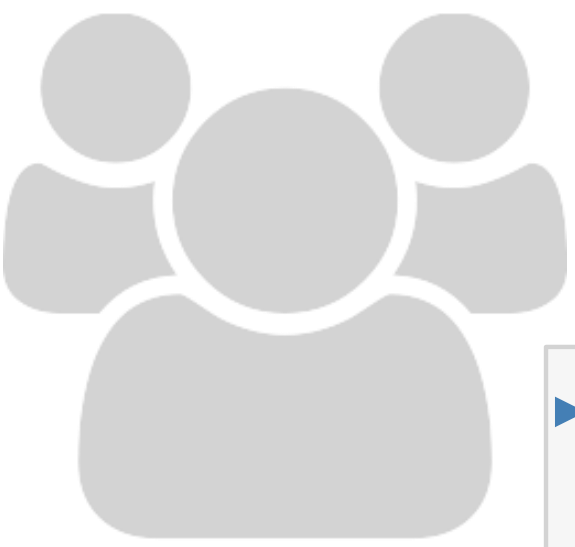
```
SELECT column_name(s), max(column_name)
FROM table_name
WHERE condition
GROUP BY column_name(s)
```

# Pop Quiz

Tidyverse equivalent?

`group_by()`

# EXERCISE



- ▶ This time we will use the Health\_SalesForce\_Contact table
- ▶ Count the number of people in each Zip Code
- ▶ Stretch Goal: Get the number by Zip Code that have received a Second Dose

**5:00**



# SOLUTION

```
SELECT
[MailingAddress.postalCode], COUNT(Id)
FROM HealthCare_DataWarehouse.ACHD.Health_SalesForce_Contact
GROUP BY [MailingAddress.postalCode]
```

```
SELECT
[MailingAddress.postalCode], COUNT(Id)
FROM HealthCare_DataWarehouse.ACHD.Health_SalesForce_Contact
WHERE Second_Dose_Received_Date__c IS NOT NULL
GROUP BY [MailingAddress.postalCode]
```



# Other Functions



# CASE

- ▶ CASE statements are for when you want to return categorical values based off of something else.

```
SELECT OrderID, Quantity,  
CASE  
    WHEN Quantity > 30 THEN "The quantity is greater than 30"  
    WHEN Quantity = 30 THEN "The quantity is 30"  
    ELSE "The quantity is under 30"  
END AS QuantityText  
FROM OrderDetails;
```

# Pop Quiz

Tidyverse equivalent?

`case_when()`

# CONCAT

- ▶ CONCAT()
  - ▶ This is mostly used when you have multiple columns you need.
  - ▶ May look different depending on DB server

```
SELECT CONCAT(column1, " ", column2) AS ConcatenatedString;
```

OR

```
SELECT column1 || " " || column2 AS ConcatenatedString;
```



# Pop Quiz

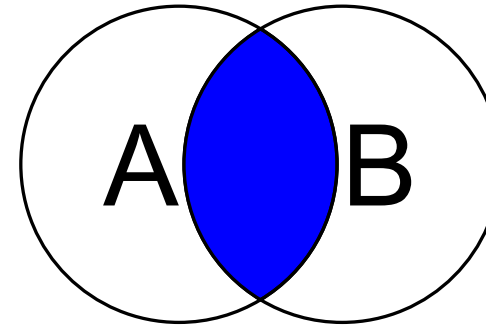
R equivalent?

`paste() / paste0()`

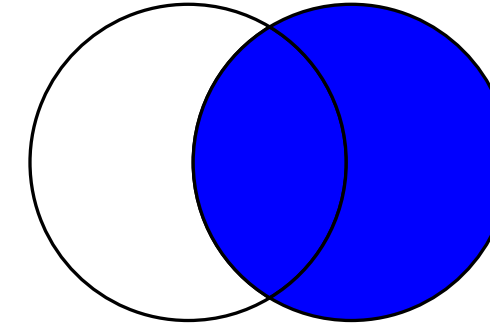
# JOINS



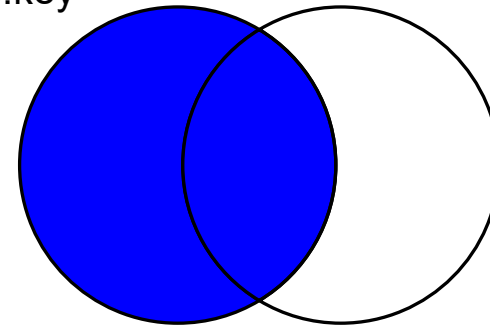
SELECT <fields>  
FROM TableA A  
INNER JOIN TableB B  
ON A.key = B.key



SELECT <fields>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.key = B.key



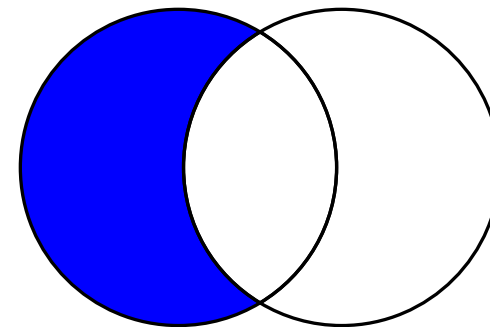
SELECT <fields>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.key = B.key



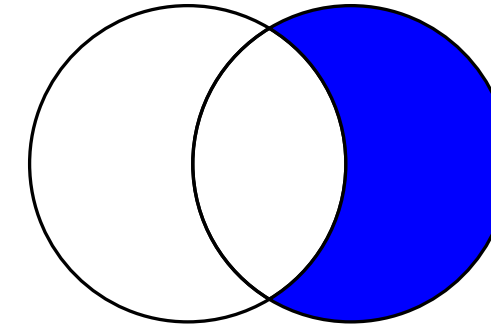
# SQL

# JOINS

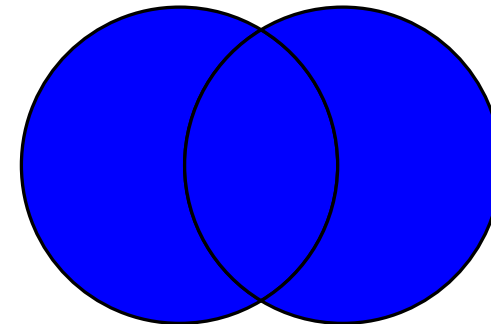
SELECT <fields>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.key = B.key  
WHERE B.key IS NULL



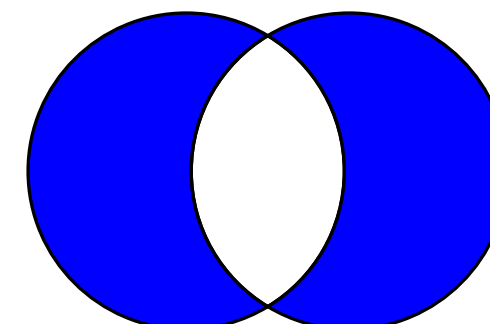
SELECT <fields>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.key = B.key  
WHERE A.key IS NULL



SELECT <fields>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.key = B.key



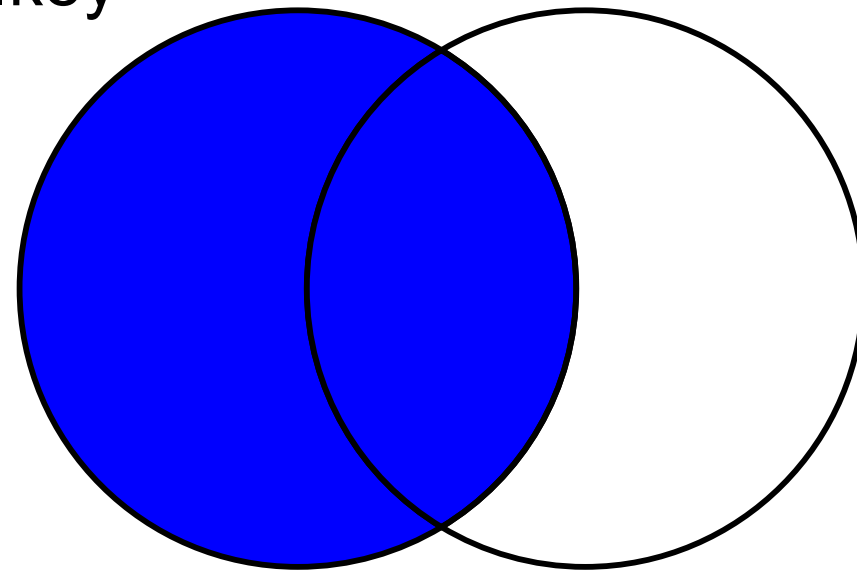
SELECT <fields>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL



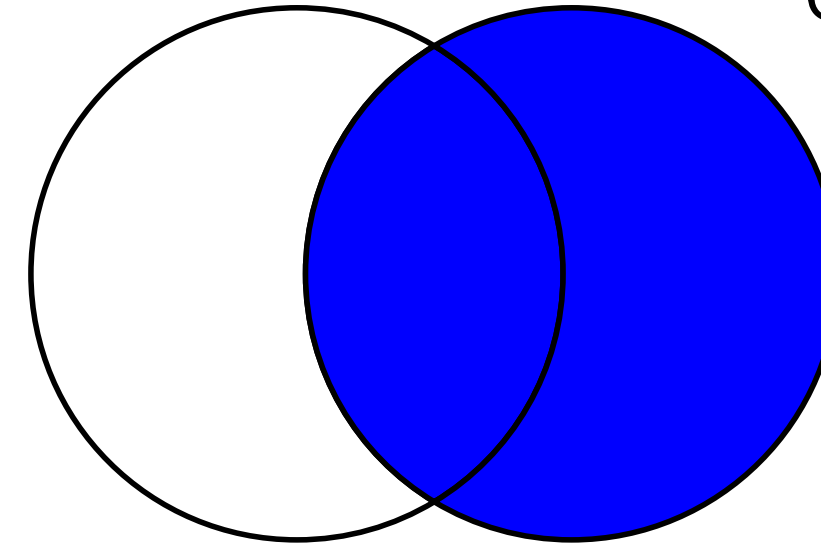
This work is licensed under a Creative Commons Attribution 3.0 Unported License.  
Author: <http://commons.wikimedia.org/wiki/User:Arbeck>

# Left/Right Join

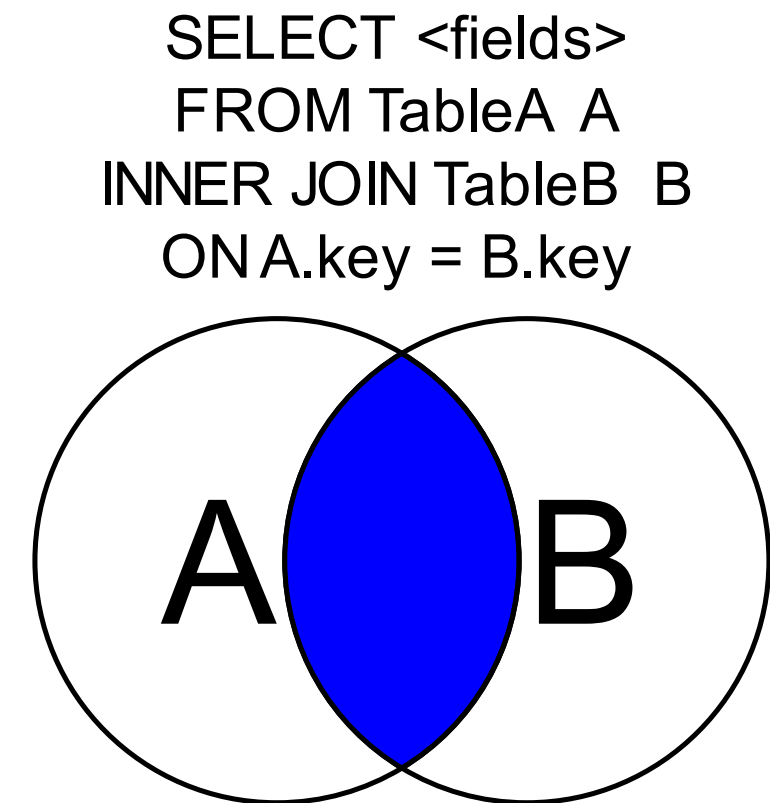
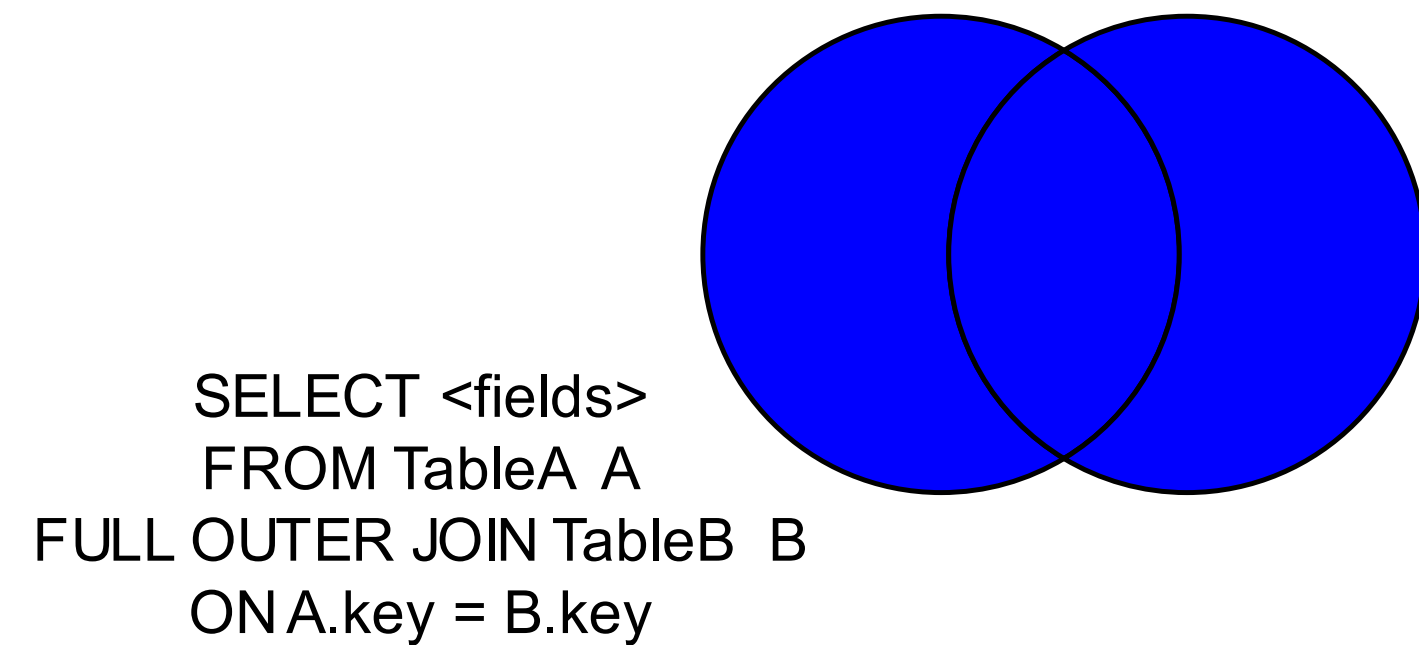
```
SELECT <fields>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.key = B.key
```



```
SELECT <fields>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.key = B.key
```



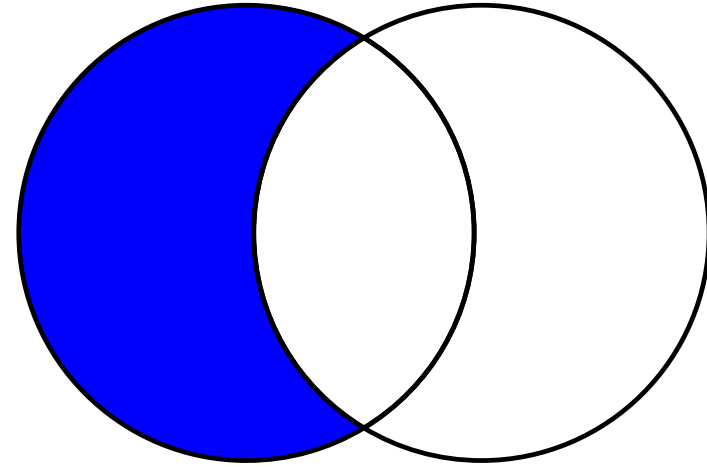
# Inner/Outer Join



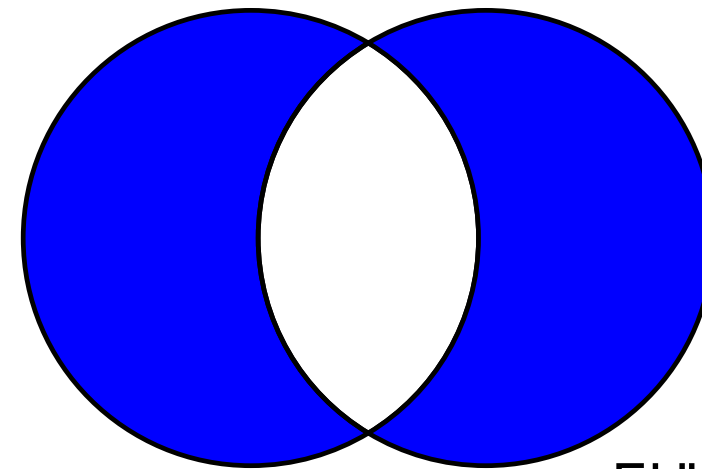
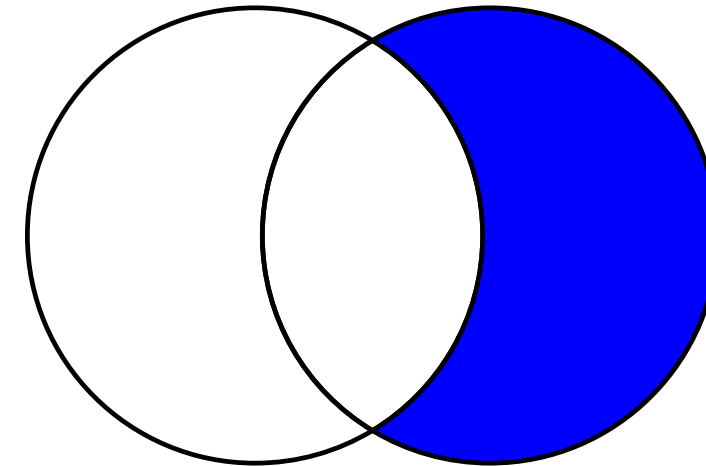


# Anti-Joins

```
SELECT <fields>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.key = B.key  
WHERE B.key IS NULL
```



```
SELECT <fields>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.key = B.key  
WHERE A.key IS NULL
```



```
SELECT <fields>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL
```

DB

Connections



# CONNECTING

- ▶ Database connectors require that your computer has the necessary software.
  - ▶ This will depend on what database type you are trying to connect to



# ALLOWING HANDSHAKES

- ▶ To setup database connections you will need to install the proper drivers.
  - ▶ The steps for this can be found here: <https://db.rstudio.com/best-practices/drivers/>
  - ▶ In general setup on Windows is a little bit easier since ODBC Data Source Administrator can be used
- ▶ Your machine may already have drives installed if you've already installed SQL IDE's such as: pgAdmin, DBeaver, or the MySQL Workbench

# Storing Credentials





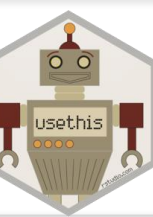
# ENVIRONMENTAL VARIABLE OR FILE

- ▶ You should never “hard code” your credentials into an app.
- ▶ Instead you should store them as environmental variables, or in a hidden file that you ignore in the Git Repository

- ▶ Why?

If something requires that you to login, we can assume that not just anybody should be able to access it.

Think of your credentials like your debit card and pin number



# BUILDING AN ENVIRON FILE

- ▶ The usethis package has a function that will build your .Renviron file in your directory or for your entire profile.

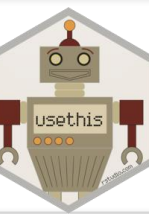
```
usethis::edit_r_environ()  
usethis::edit_r_environ("project")
```

- ▶ How are .Renviron Files structured?

```
uid=some_username  
pwd=Password
```

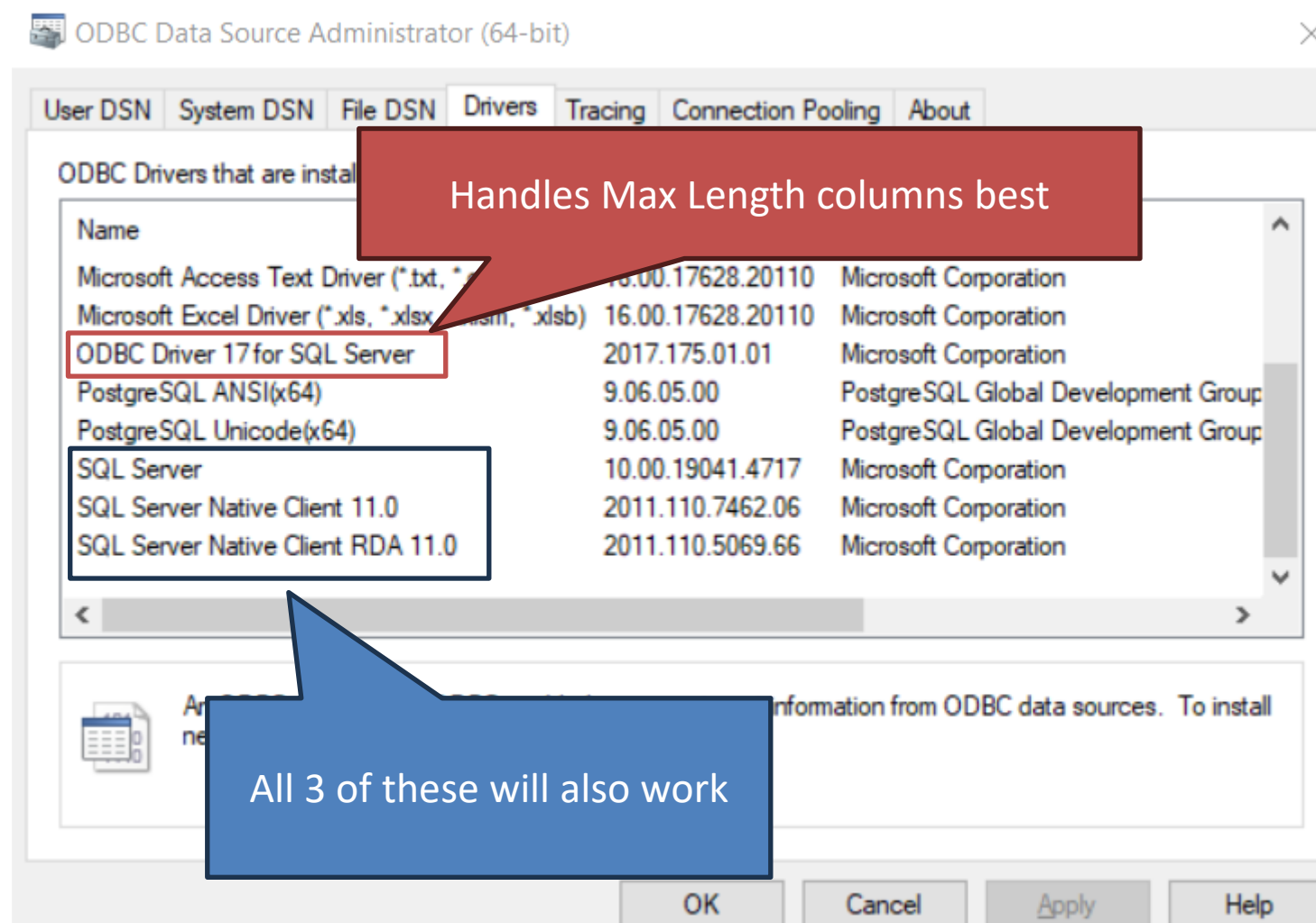
A new line for each variable

No spaces between variable name and value

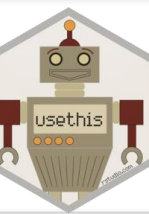


# SQL Drivers

- ▶ Checking Installed Drivers
  - ▶ Open ODBC Data Source Administrator (64-bit)
  - ▶ Ask DIT to install the one you need if it's missing



driver = "{ODBC Driver 17 for SQL Server}"



# LOADING VARIABLES

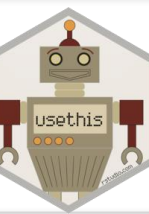
- ▶ Small difference between credentials in your profile or the project folder.
- ▶ The string argument is the name you gave your variable

## Profile

```
Sys.getenv("uid")  
Sys.getenv("pwd")
```

## .Renviron

```
readRenviron(".Renviron")  
  
Sys.getenv("uid")  
Sys.getenv("pwd")
```



# ESTABLISHING CONNECTIONS

- ▶ Each data base type has a different connection string and list of requirements.

```
conn <- dbConnect(odbc::odbc(), driver = "{Driver Name}", server =  
  "IP_or_HOST_ADDRESS", port = port#, database = "DBName", uid = un, pwd = pwd)
```

- ▶ More on connection strings: <https://db.rstudio.com/best-practices/drivers/#connecting-to-a-database-in-r>

# Running a Query

- ▶ If you want to load the whole table:

```
tbl <- dbReadTable(con, SQL("Schema.TableNameHere"))
```

- ▶ If you want to run a custom query:

```
tbl <- dbReadTable(con, "Text of your SQL Query")
```



# Your Turn

Go to Rstudio and open the `dbi_example.Rmd` and follow instructions and run the code chunks up to disconnect.

