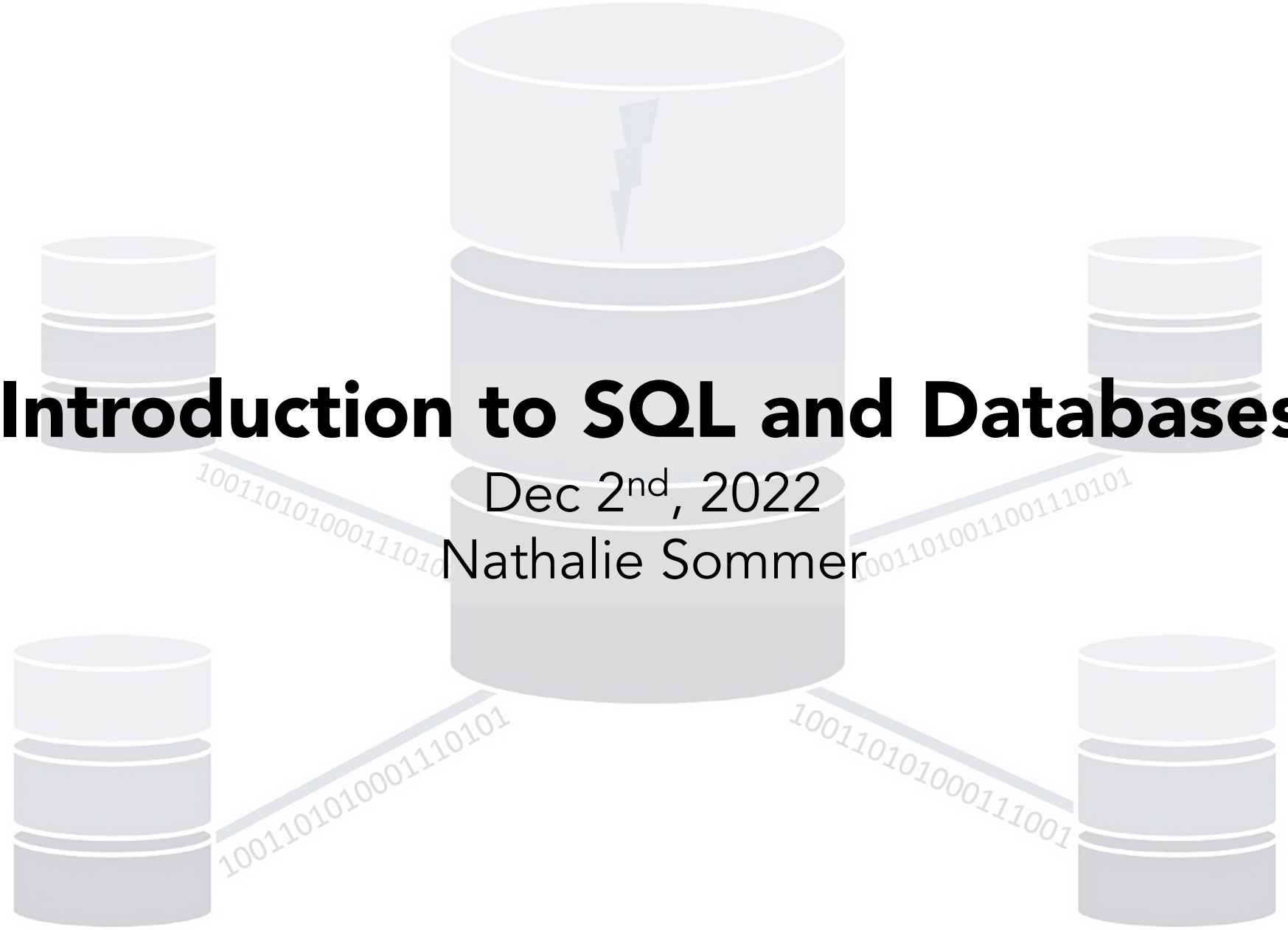
SCOPE SIG + YEDSI

SPRY: A Learning Community for Quantitative Skill-Sharing

Slack Channel: bit.ly/3TqANgR

Listserv: Contact nathalie.sommer@yale.edu

Got an idea for a future event? Drop us a message.



Introduction to SQL and Databases

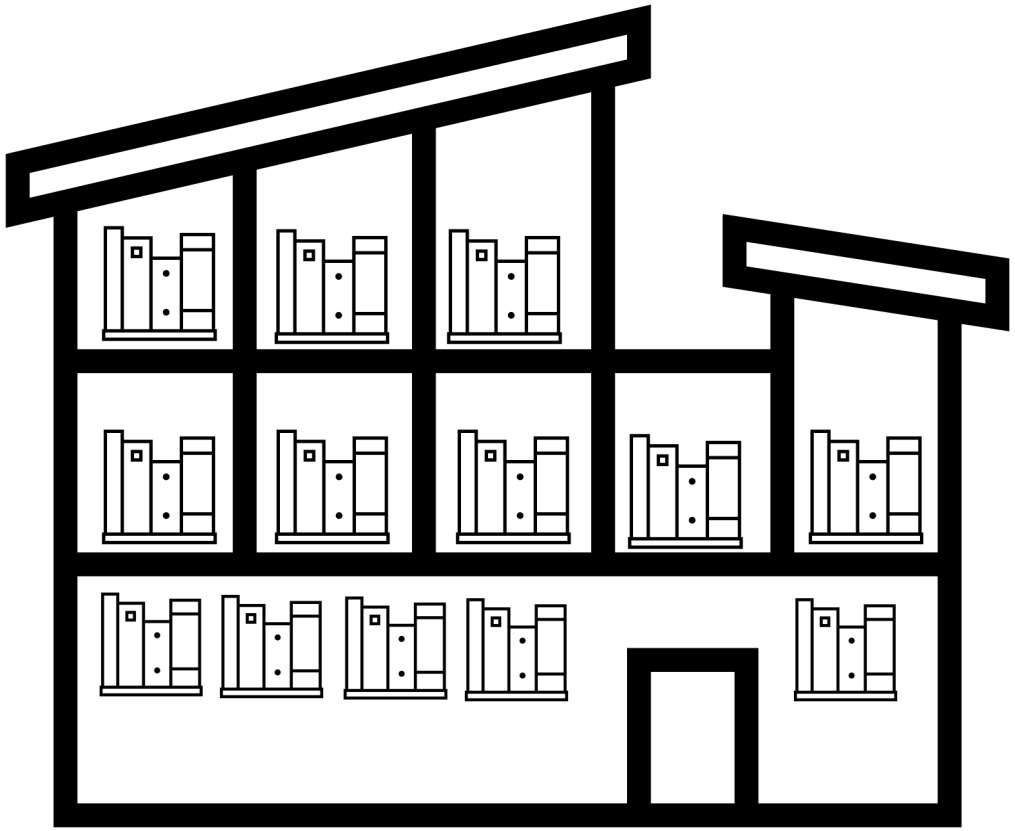
Dec 2nd, 2022

Nathalie Sommer

Objectives

- Motivation for databases and SQL in research
- Types of databases
- Overview of SQL syntax
- Tutorial

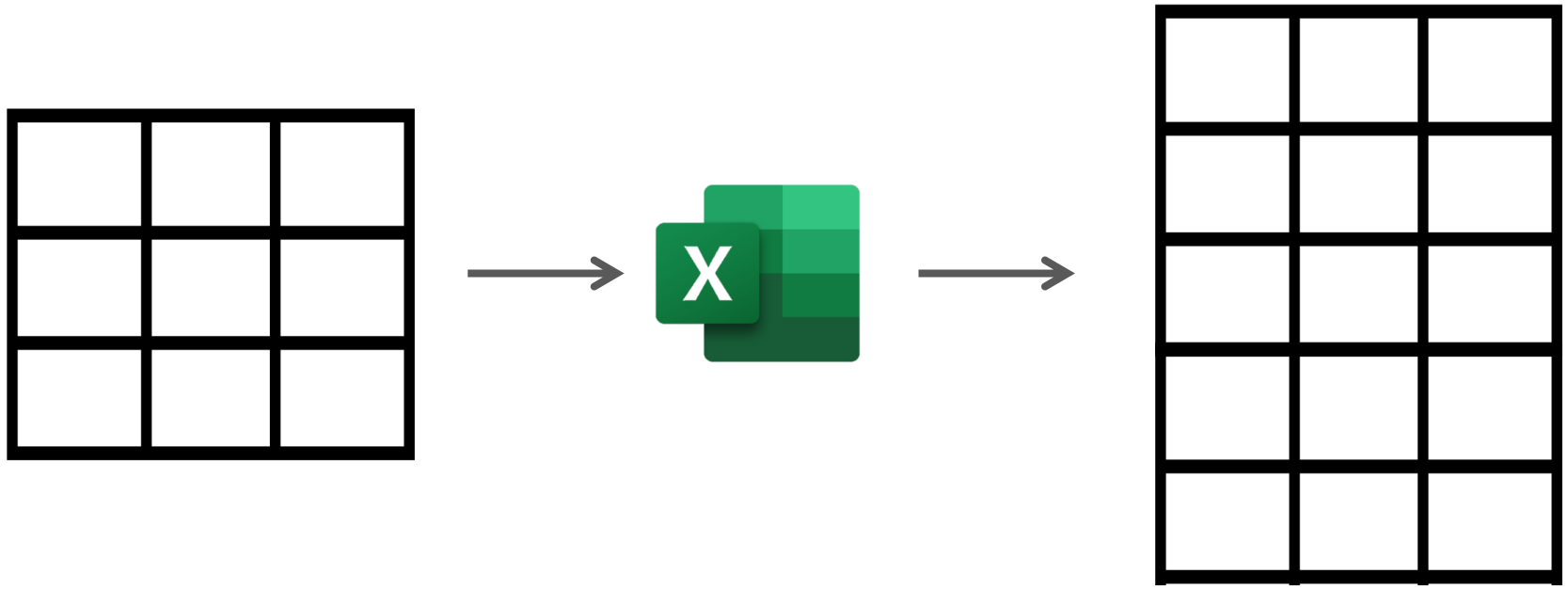
What is a database?



What is SQL?

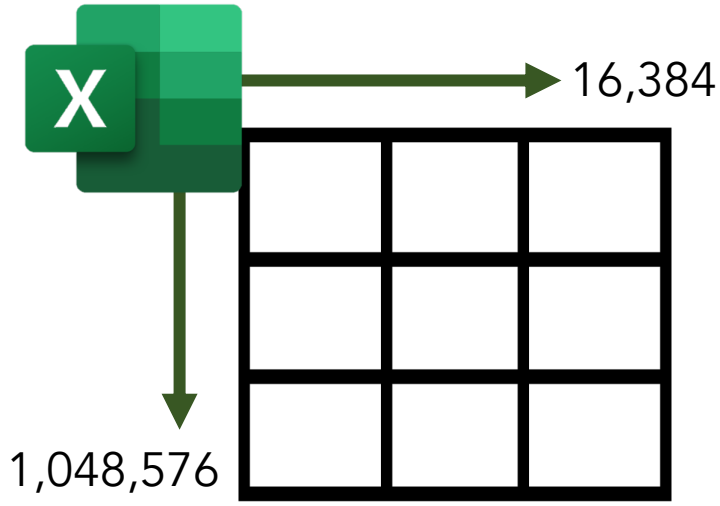
Why use databases in research?

Current approach: Flat files



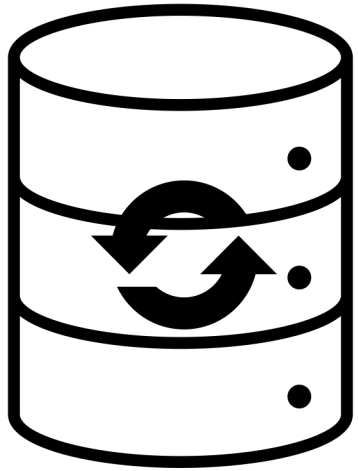
Why use databases in research?

Size



Security

Accuracy

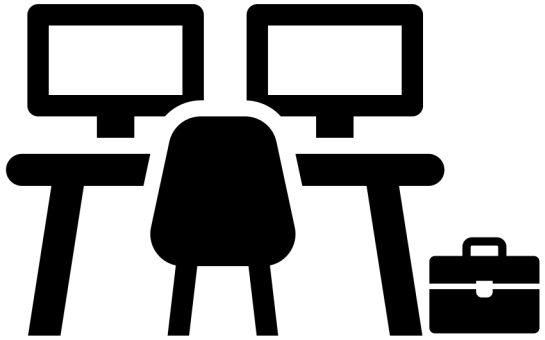


Why learn SQL?

Personal database

Public data access

In-demand skill



Databases and SQL in Academia

Lab management

Long term projects

"Protects" data

Database Management Systems

- Hierarchical: one-to-many, tree structure
- Network: many-to-many, branching
- Object-oriented: like network, but with different data “types”
- Relational: can do it all!

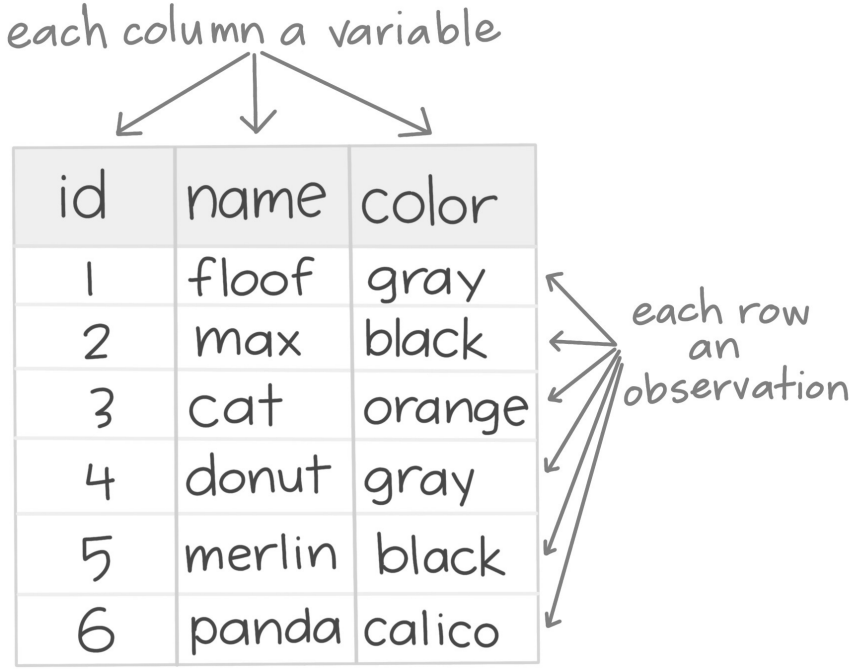
Illustration by Julie Lowndes and Allison Horst

"TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."

–HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement



Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

Vendor	City	Delivery	Rating
One6Three	New Haven	Phone	Best
Est Est Est	New Haven	Online	Better
BAR	New Haven	None	Good
Modern	New Haven	None	Superior

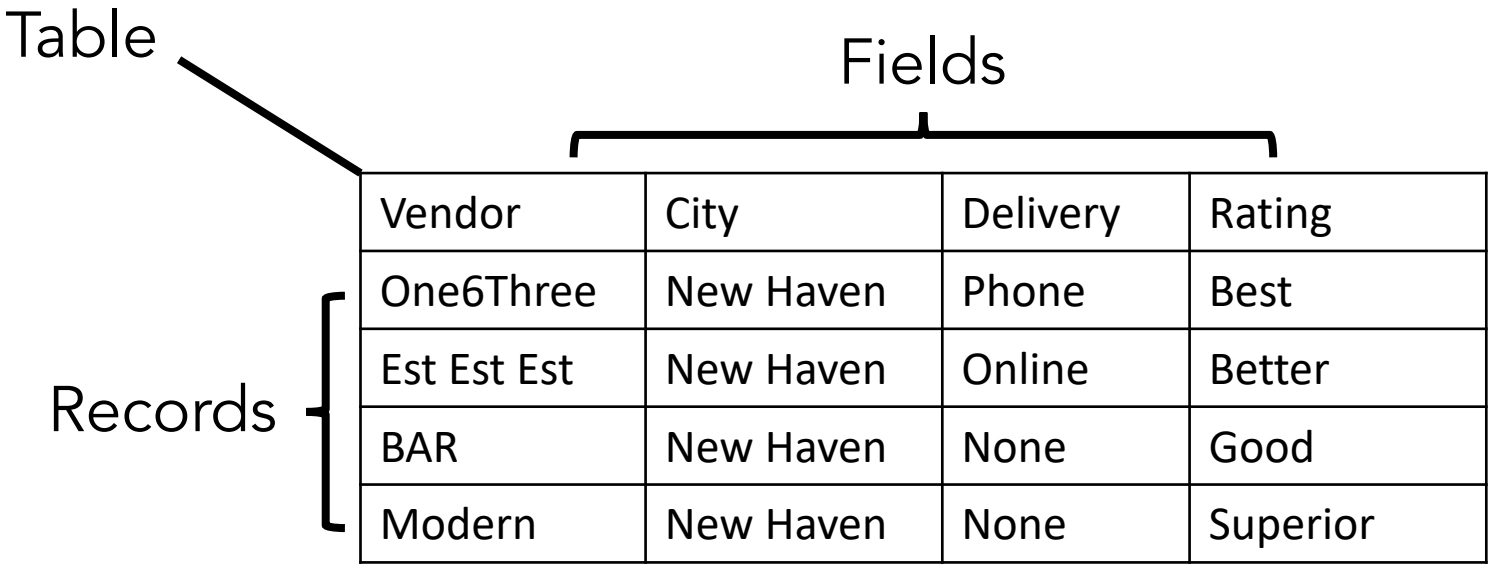
Vendor	Cheese	Crust	Toppings
One6Three	Good	Good	Excellent
Est Est Est	Good	Fair	Good
BAR	Excellent	Fair	Good
Modern	Good	Excellent	Excellent

Vendor	City	Delivery	Rating
One6Three	New Haven	Phone	Best
Est Est Est	New Haven	Online	Better
BAR	New Haven	None	Good
Modern	New Haven	None	Superior

Vendor	Cheese	Crust	Toppings
One6Three	Good	Good	Excellent
Est Est Est	Good	Fair	Good
BAR	Excellent	Fair	Good
Modern	Good	Excellent	Excellent

Vendor	City	Delivery	RatingType	Rating
One6Three	New Haven	Phone	Overall	Best
One6Three	New Haven	Phone	Cheese	Good
One6Three	New Haven	Phone	Crust	Good
One6Three	New Haven	Phone	Toppings	Excellent
Est Est Est	New Haven	Online	Overall	Better
Est Est Est	New Haven	Online	Cheese	Good
Est Est Est	New Haven	Online	Crust	Fair
Est Est Est	New Haven	Online	Toppings	Good
BAR	New Haven	None	Overall	Good
BAR	New Haven	None	Cheese	Excellent
BAR	New Haven	None	Crust	Fair
BAR	New Haven	None	Toppings	Good
Modern	New Haven	None	Overall	Superior
Modern	New Haven	None	Cheese	Good
Modern	New Haven	None	Crust	Excellent
Modern	New Haven	None	Toppings	Excellent

Relational Databases (RDBMS)



Vendor	City	Delivery	Rating
One6Three	New Haven	Phone	Best
Est Est Est	New Haven	Online	Better
BAR	New Haven	None	Good
Modern	New Haven	None	Superior

Relational Databases (RDBMS)

Primary Key

Foreign Key

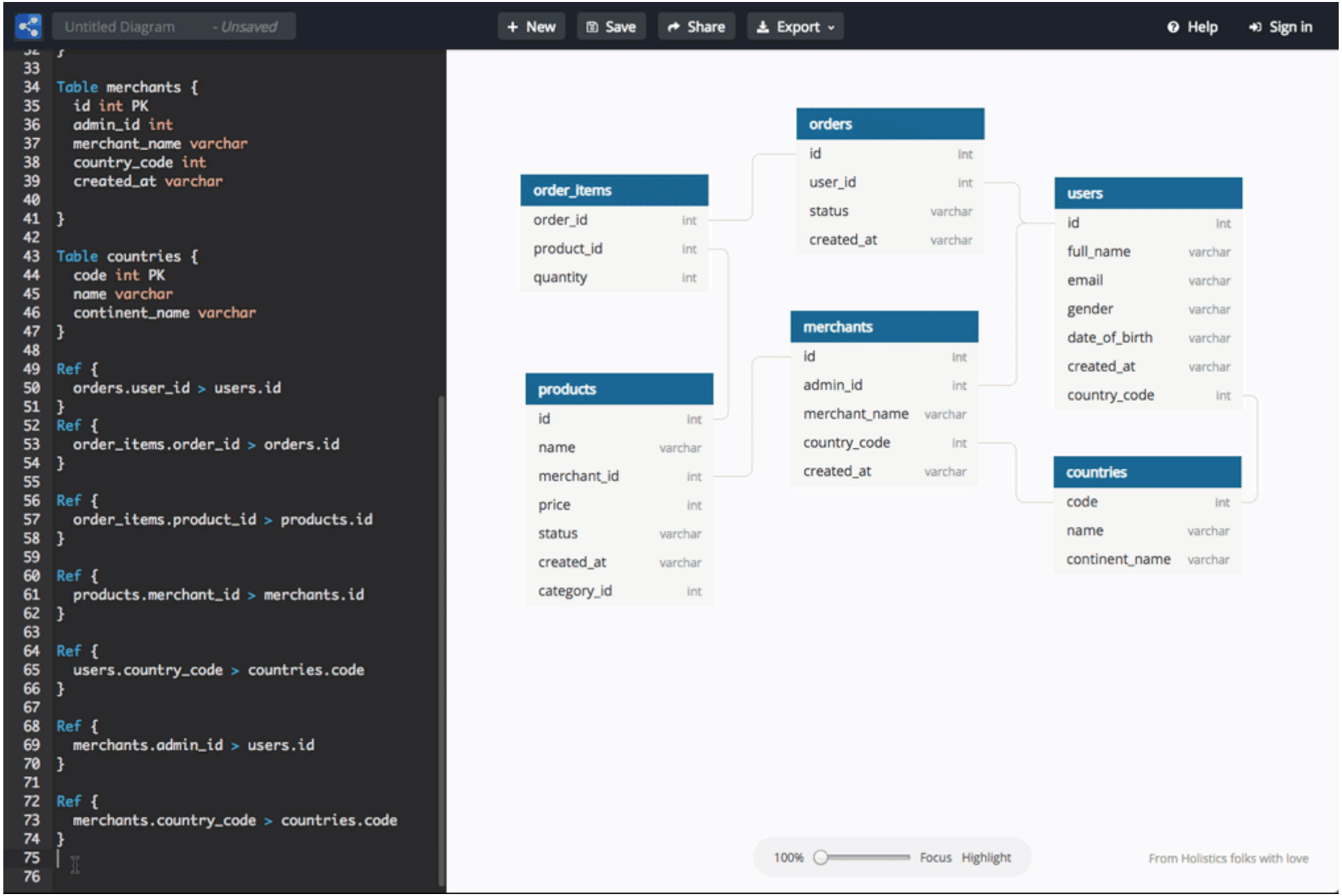
Vendor	City	Delivery	Rating
One6Three	New Haven	Phone	Best
Est Est Est	New Haven	Online	Better
BAR	New Haven	None	Good
Modern	New Haven	None	Superior

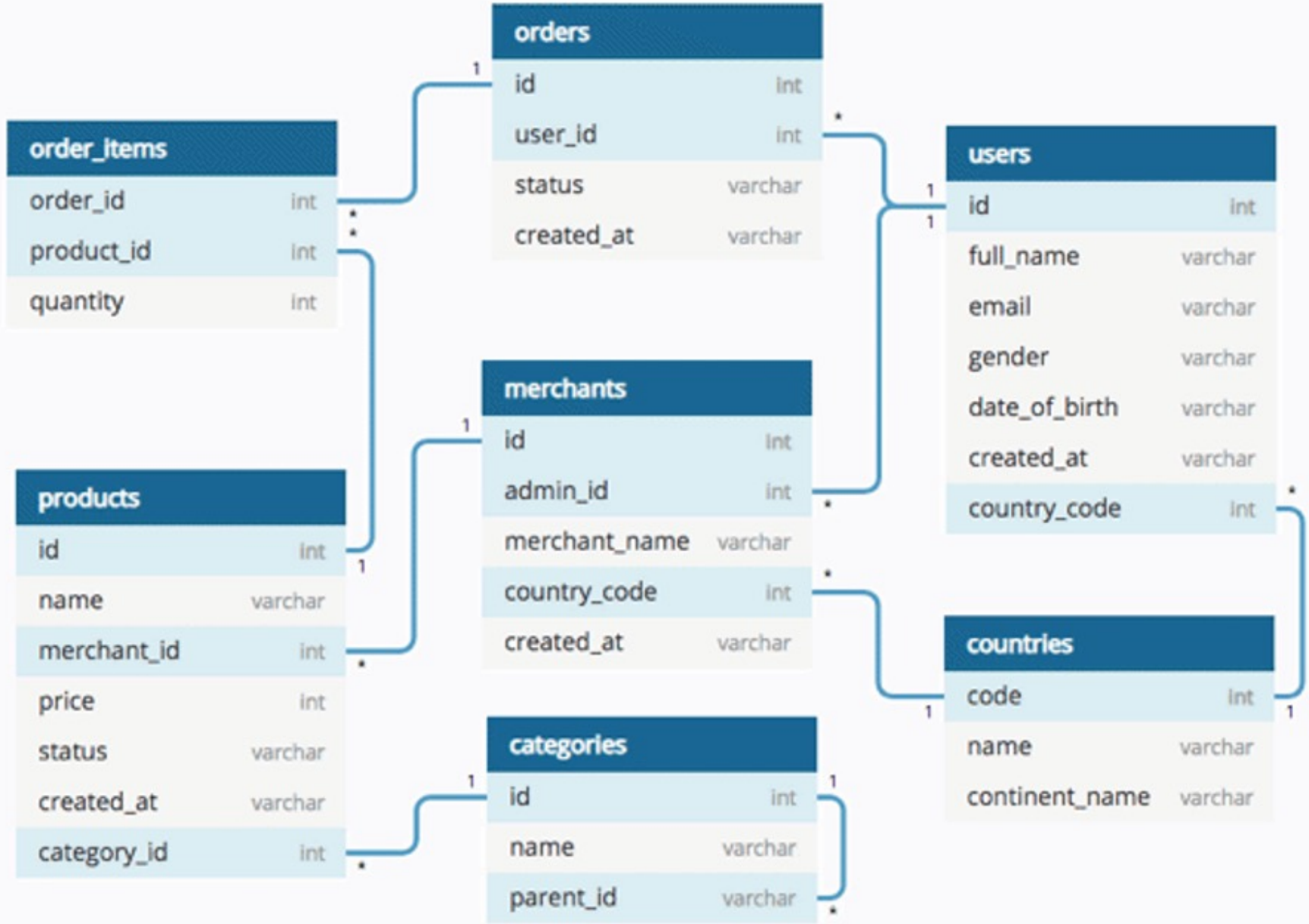
Cheese	Crust	Toppings
Good	Good	Excellent
Good	Fair	Good
Excellent	Fair	Good
Good	Excellent	Excellent

Relational Databases (RDBMS)

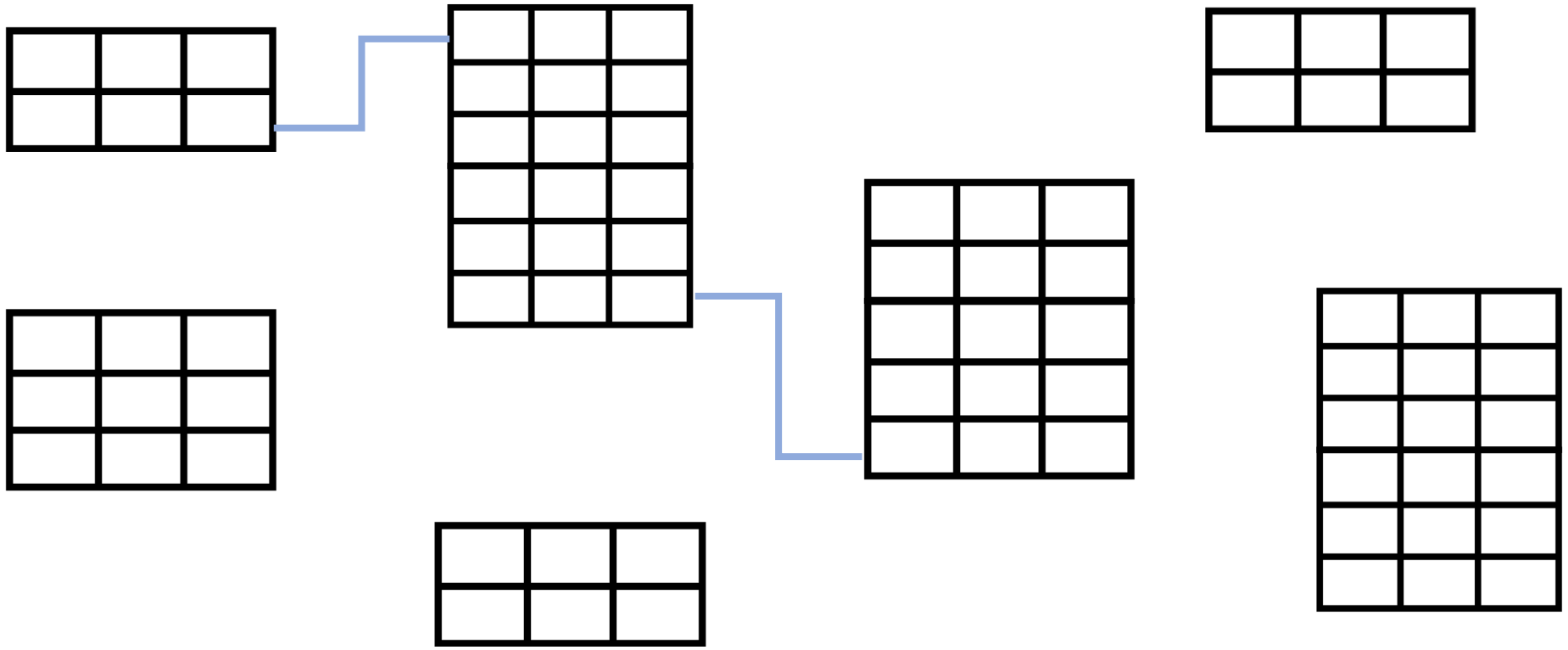
Join

Vendor	City	Delivery	Rating	PK	FK	Cheese	Crust	Toppings
One6Three	New Haven	Phone	Best	1	1	Good	Good	Excellent
Est Est Est	New Haven	Online	Better	2	2	Good	Fair	Good
BAR	New Haven	None	Good	3	3	Excellent	Fair	Good
Modern	New Haven	None	Superior	4	4	Good	Excellent	Excellent


Untitled Diagram - Unsaved
+ New
Save
Share
Export
Help
Sign in
Table merchants {
id int PK
admin_id int
merchant_name varchar
country_code int
created_at varchar
}
Table countries {
code int PK
name varchar
continent_name varchar
}
Ref {
orders.user_id > users.id
}
Ref {
order_items.order_id > orders.id
}
Ref {
order_items.product_id > products.id
}
Ref {
products.merchant_id > merchants.id
}
Ref {
users.country_code > countries.code
}
Ref {
merchants.admin_id > users.id
}
Ref {
merchants.country_code > countries.code
}
orders
id int
user_id int
status varchar
created_at varchar
order_items
order_id int
product_id int
quantity int
users
id int
full_name varchar
email varchar
gender varchar
date_of_birth varchar
created_at varchar
country_code int
merchants
id int
admin_id int
merchant_name varchar
country_code int
created_at varchar
products
id int
name varchar
merchant_id int
price int
status varchar
created_at varchar
category_id int
countries
code int
name varchar
continent_name varchar
100%
Focus
Highlight
From Holistics folks with love


orders
id int
user_id int
status varchar
created_at varchar
order_items
order_id int
product_id int
quantity int
users
id int
full_name varchar
email varchar
gender varchar
date_of_birth varchar
created_at varchar
country_code int
merchants
id int
admin_id int
merchant_name varchar
country_code int
created_at varchar
products
id int
name varchar
merchant_id int
price int
status varchar
created_at varchar
category_id int
countries
code int
name varchar
continent_name varchar
categories
id int
name varchar
parent_id varchar

Relational Databases (RDBMS)



Types of RDBMS

Dialects of SQL

SQL

- **S**tructured **Q**uery **L**anguage, a *declarative programming language*

1. Storing and organizing data
2. Retrieving and cleaning data

- Not the "database" itself
- Not a data visualization tool
- Not used for complex data analysis
- Used to select, import, and aggregate data from databases

SQL Syntax

Declarative language

Verb + Subject

```
SELECT delivery, vendor
FROM pizza
```

Vendor	City	Delivery	Rating
One6Three	New Haven	Phone	Best
Est Est Est	New Haven	Online	Better
BAR	New Haven	None	Good
Modern	New Haven	None	Superior

SQL Syntax

Declarative language

Verb + Subject, with optional predicate

```
SELECT delivery, vendor
FROM pizza
WHERE
rating = “Superior”;
```

Vendor	City	Delivery	Rating
One6Three	New Haven	Phone	Best
Est Est Est	New Haven	Online	Better
BAR	New Haven	None	Good
Modern	New Haven	None	Superior

SQL Syntax

`SELECT`: Chooses the field(s)

`FROM`: Specifies the table in the database

`WHERE`: Conditional filtering for records

`JOIN`: Retrieves and links tables based on a key or field ID

`GROUP BY`: Aggregates records

`HAVING`: Filters aggregated groups.

`ORDER BY`: Orders or sorts results.

`LIMIT`: Limits results to a certain number of records

Tutorial

1. Download SQLite: https://sqlite.org/download.html
2. Create a local folder to work in
3. Download and save these files in your local folder: https://tinyurl.com/SPRY-SQL
4. Open sqlite3


dbplyr
dbplyr.tidyverse.org

Same concept, different syntax

dplyr

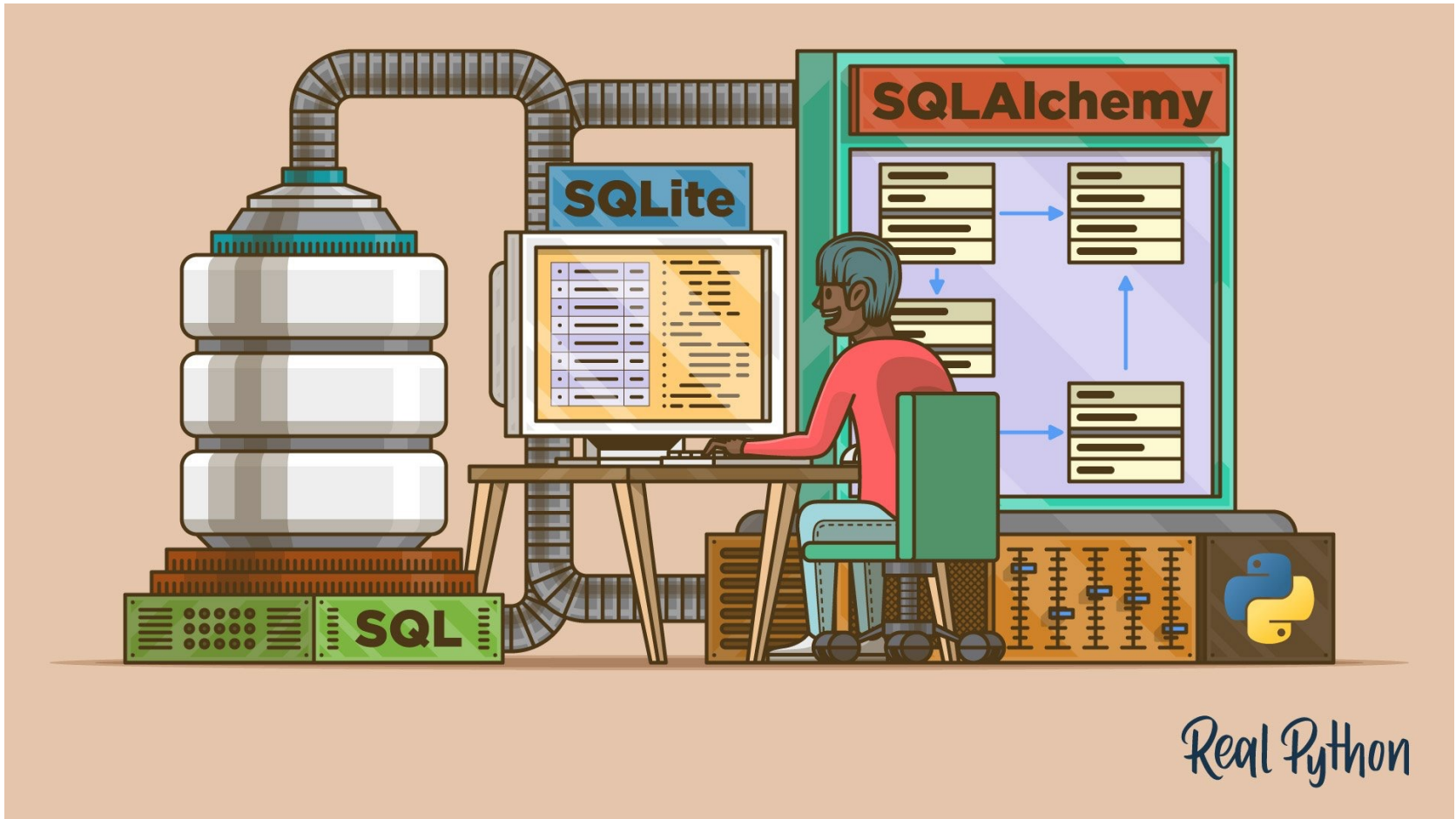
```
table %>%
  filter(field == "value") %>%
  left_join(lkup,
    by = c("lkup_id" = "id") %>%
  group_by(year) %>%
  summarize(N = sum(1)) %>%
  filter(N > 100) %>%
  arrange(desc(N)) %>%
  head(10)
```

MySQL

```
SELECT
  year, sum(1) as N
FROM table t
LEFT JOIN lkup l
  ON t.lkup_id = l.id
WHERE field = "value"
GROUP BY year
HAVING N > 100
ORDER BY N desc
LIMIT 0, 10;
```

Credit: Data Carpentries

SQLAlchemy



Tutorials

- UVA Data Library: https://data.library.virginia.edu/creating-a-sqlite-database-for-use-with-r/
- Data Carpentries: https://datacarpentry.org/R-ecology-lesson/05-r-and-databases.html
- Plus, thousands more on the World Wide Web

SCOPE SIG + YEDSI

SPRY: A Learning Community for Quantitative Skill-Sharing

Slack Channel: bit.ly/3TqANgR

Listserv: Contact nathalie.sommer@yale.edu

Got an idea for a future event? Drop us a message.

Up Next:

December 9th

Alt-ac data careers

Sage 32

Andis Arietta, PhD