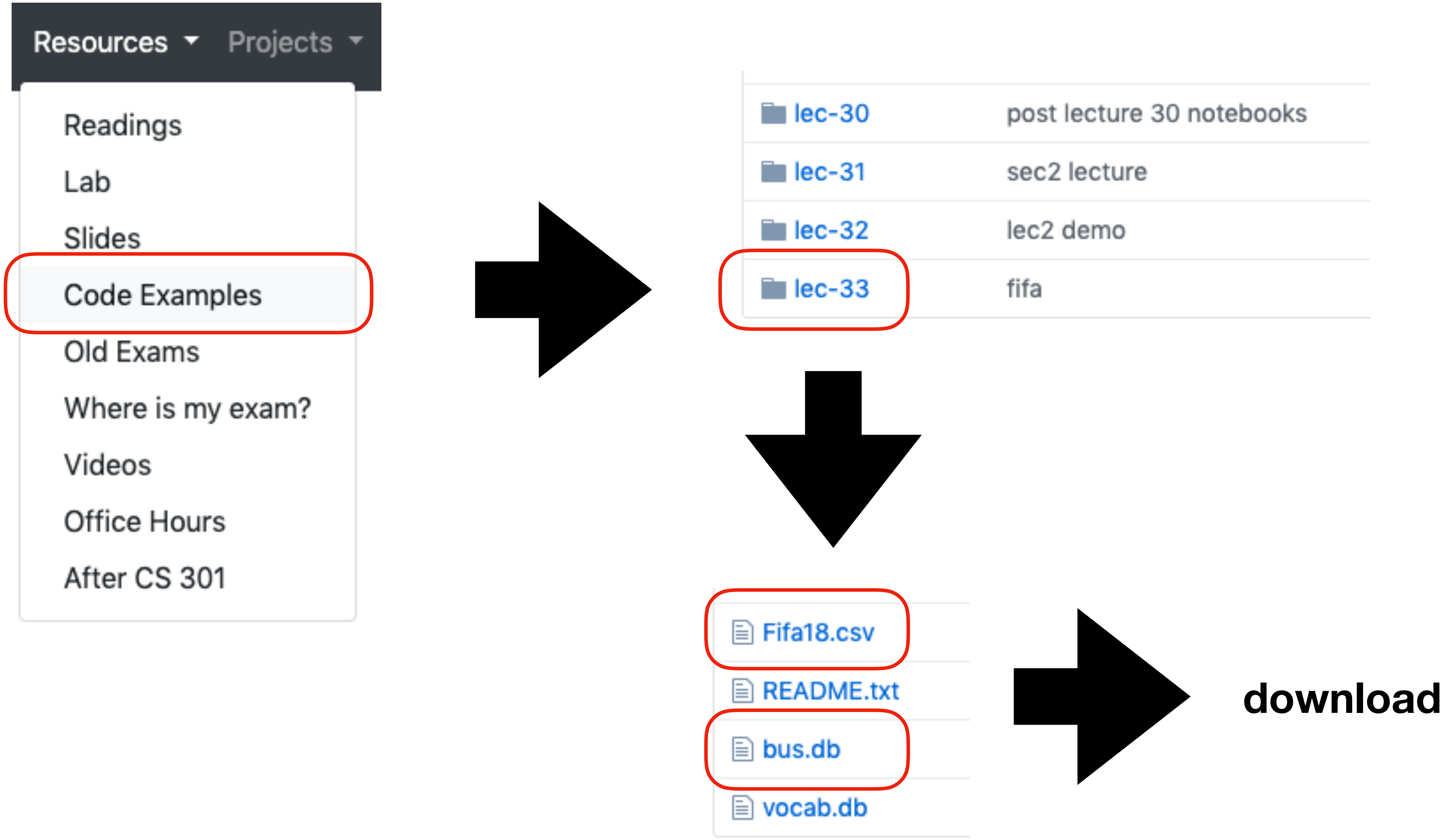
[301] Database 1

Tyler Caraza-Harter

To download...



lec-30	post lecture 30 notebooks
lec-31	sec2 lecture
lec-32	lec2 demo
lec-33	fifa

Fifa18.csv

README.txt

bus.db

vocab.db

download

301 Progress

Languages learned

- Python [**Programming** Language]
- HTML [**Markup** Language]

Data storage

- CSV files
- JSON files

301 Progress

Languages learned

- Python [**Programming** Language]
- HTML [**Markup** Language]
- SQL [**Query** Language]

Data storage

- CSV files
- JSON files
- SQL databases

structured query language

Learning Objectives Today

SQL Data

- schemas: tables, columns, types
- advantages over JSON/CSV

SQL Queries

- select, where, limit, sort by
- sqlite3 module
- Pandas/DB integration

Outline

Tabular Data: CSVs vs. Databases

Common SQL Databases

Example: Madison bus-route data

SQL: Structured Query Language

Querying from Python

CSV

State	Capital	Population	Area
WI	Madison	5795000	65498
…	…	…	…

Characteristics

- one table

SQL Database

capitals

State	Capital
WI	Madison
…	…

populations

State	Population
WI	5795000
…	…

counties

County	Pop	un_emp
Dane	536416	0.02
…	…	…

areas

State	Area
WI	65498
…	…

Characteristics

- collection of tables, each named

CSV

State	Capital	Population	Area
WI	Madison	5795000	65498
…	…	…	…

Characteristics

- one table
- columns *sometimes* named

SQL Database

capitals

State	Capital
WI	Madison
…	…

populations

State	Population
WI	5795000
…	…

counties

County	Pop	un_emp
Dane	536416	0.02
…	…	…

areas

State	Area
WI	65498
…	…

Characteristics

- collection of tables, each named
- columns *always* named

CSV

State	Capital	Population	Area
string	string	string	string
string	string	string	string
string	string	string	string
string	string	string	string
string	string	string	string
string	string	string	string
string	string	string	string

Characteristics

- one table
- columns *sometimes* named
- everything is a string

SQL Database

capitals

State	Capital
text	text
text	text
text	text
text	text

populations

State	Population
text	integer
text	integer
text	integer
text	integer

counties

County	Pop	un_emp
text	integer	real
text	integer	real
text	integer	real
text	integer	real

areas

State	Area
text	integer
text	integer
text	integer
text	integer

no text allowed

Characteristics

- collection of tables, each named
- columns *always* named
- types per column (enforced)

Why use a database?

1. More Structure

Database

A	B	C
text	integer	real
text	integer	real
text	integer	real
text	integer	real

same fields and same types in every column

CSV

```
A,B,C
string,string,string
string,string,string
string,string,string
string,string,string
```

everything is a string

JSON

```
[{"A":"val", "B":10, "C":3.14},
{"A":"val"},
{"A":"v2", "B": 9, "C":False},
```

types, but…

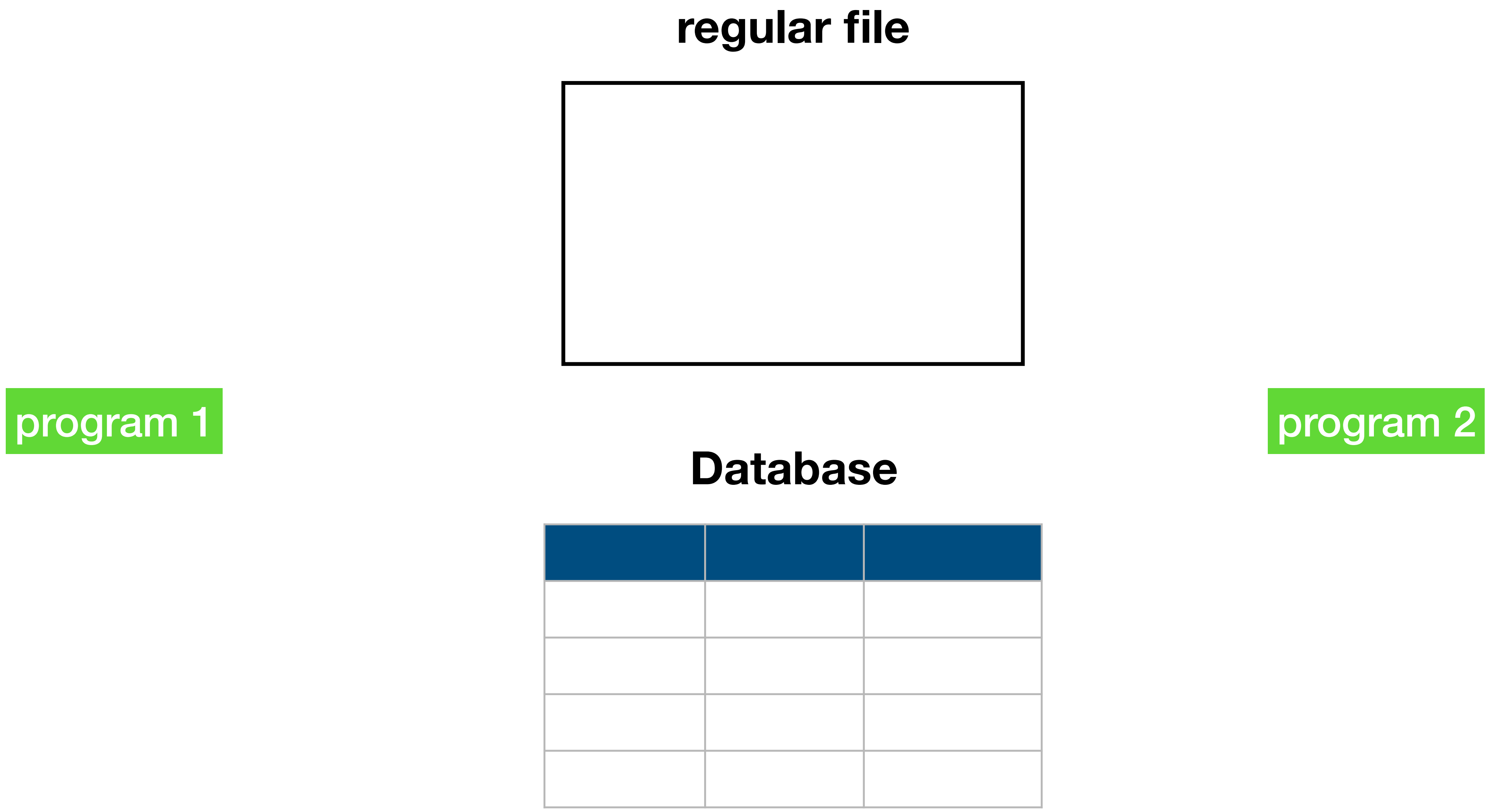
missing values

types may differ across columns

Why use a database?

1. More Structure

2. Sharing



regular file

program 1

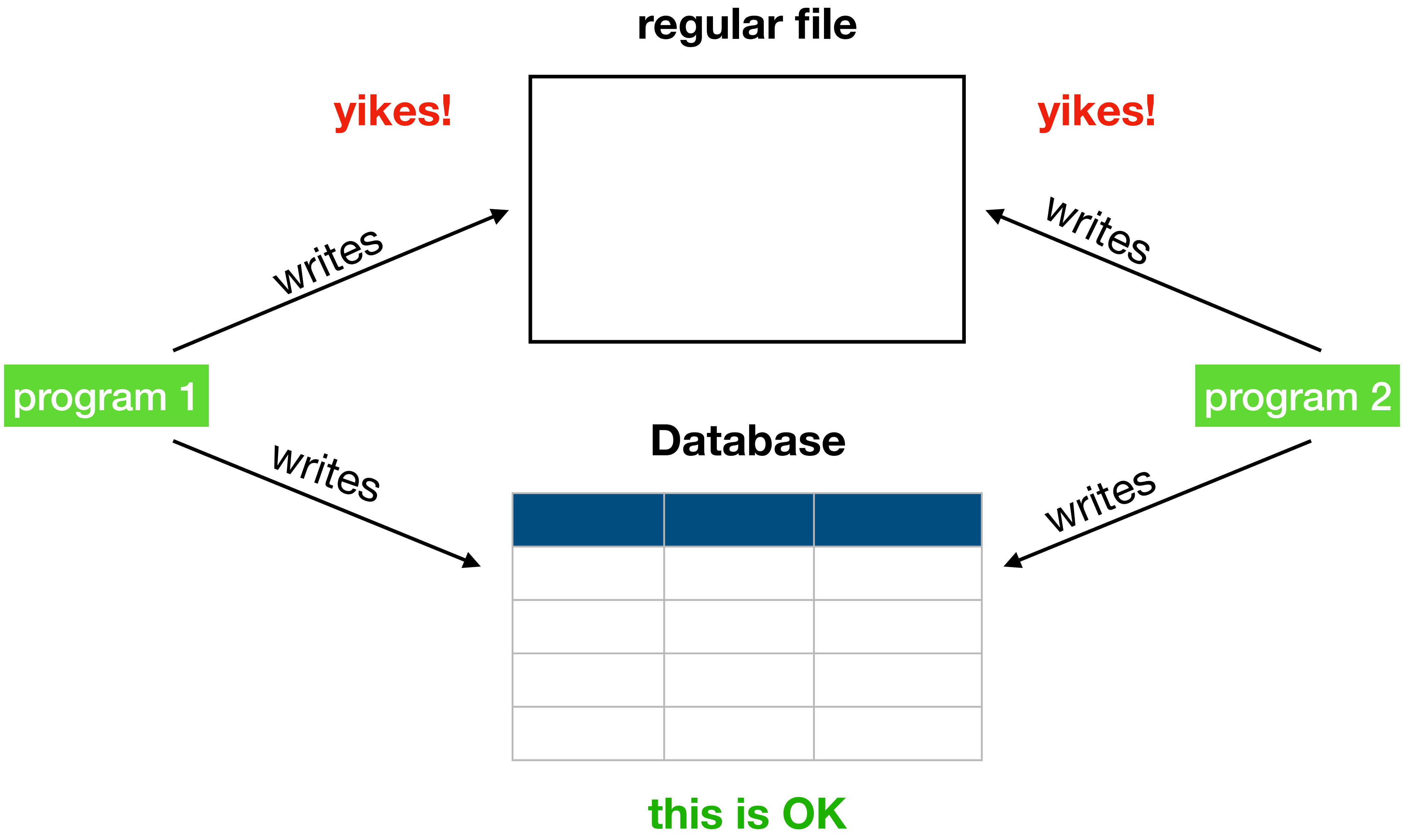
program 2

Database

Why use a database?

1. More Structure

2. Sharing



regular file

yikes!

yikes!

writes

writes

program 1

program 2

Database

writes

writes

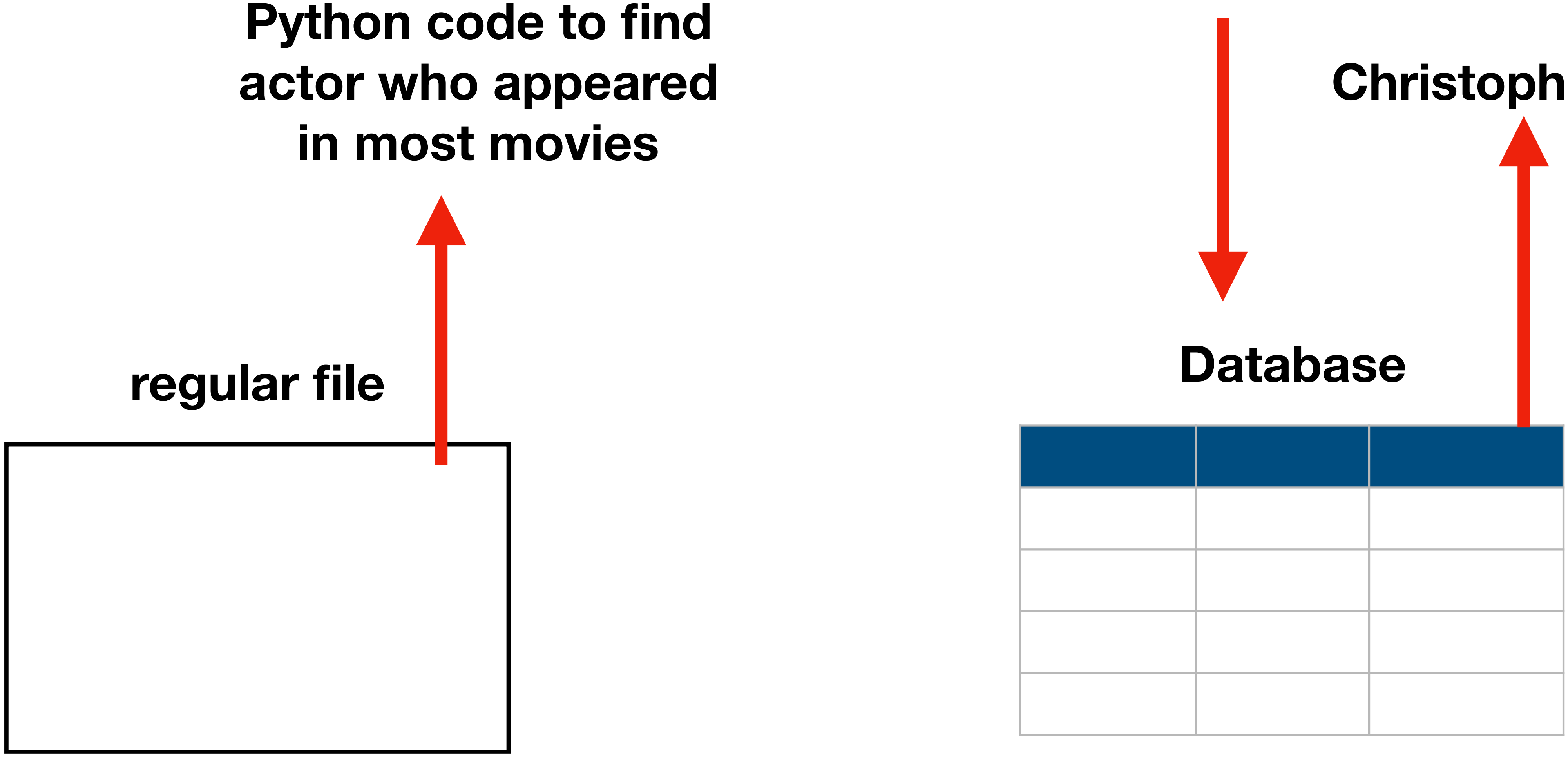
this is OK

Why use a database?

1. More Structure

2. Sharing

3. Queries



which actor appeared in the most movies?

Python code to find actor who appeared in most movies

Christopher Lee

regular file

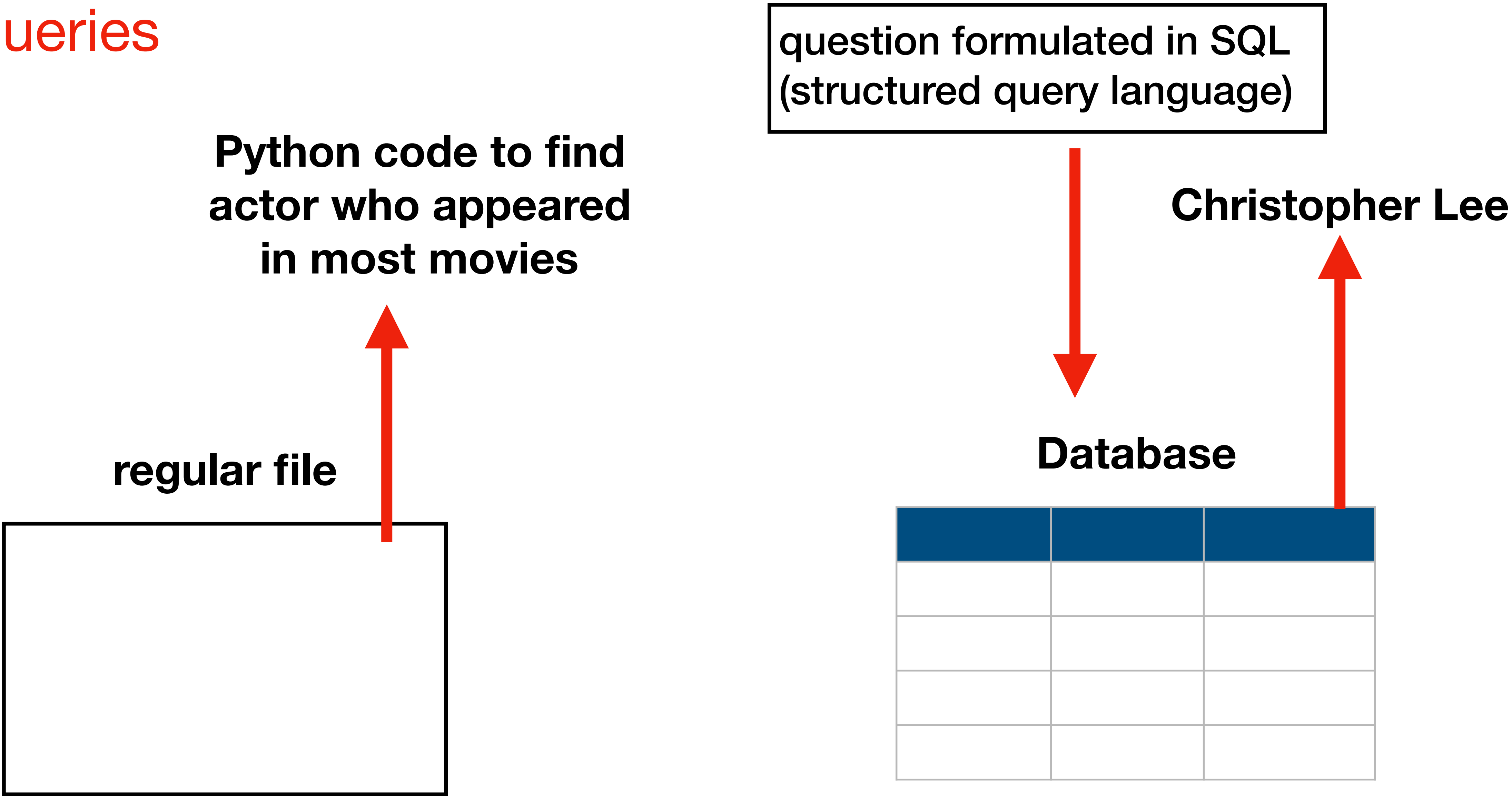
Database

Why use a database?

1. More Structure

2. Sharing

3. Queries



Python code to find actor who appeared in most movies

regular file

question formulated in SQL (structured query language)

Christopher Lee

Database

Why use a database?

1. More Structure
2. Sharing
3. Queries
4. Performance

Let's play a game where we pretend to be a database!

Question 1:

How many people are 23 or younger?

Question 2:

How many people scored 23 or less?

names	age	score
Parker	?	?
Heidy	?	?
Shirly	?	?
Arla	?	?
Bella	?	?
Bill	?	?
Hollis	?	?
Maurita	?	?
Milda	?	?
Pearline	?	?
Teresa	?	?
Ceola	?	?
Milford	?	?
Alisha	?	?
Antonetta	?	?
Ryan	?	?
Karma	?	?
Lashandra	?	?
Breana	?	?
Sara	?	?

Question 1:

How many people are 23 or younger?

Question 2:

How many people scored 23 or less?

names	age	score
Parker	26	21
Heidy	22	22
Shirly	27	22
Arla	21	22
Bella	22	22
Bill	28	22
Hollis	26	23
Maurita	22	24
Milda	22	25
Pearline	29	25
Teresa	25	25
Ceola	30	26
Milford	25	26
Alisha	30	27
Antonetta	28	28
Ryan	25	28
Karma	23	28
Lashandra	24	29
Breana	22	30
Sara	28	30

Question 1:

How many people are 23 or younger?

Question 2:

How many people scored 23 or less?

Which question took longer to answer? Why?

names	age	score
Parker	26	21
Heidy	22	22
Shirly	27	22
Arla	21	22
Bella	22	22
Bill	28	22
Hollis	26	23
Maurita	22	24
Milda	22	25
Pearline	29	25
Teresa	25	25
Ceola	30	26
Milford	25	26
Alisha	30	27
Antonetta	28	28
Ryan	25	28
Karma	23	28
Lashandra	24	29
Breana	22	30
Sara	28	30

DBs can keep multiple copies of the same data

- which organizations to use are **configured** (indexing)
- which copy to use is used is **automatically determined** based on the question being asked

names	age	score
Arla	21	22
Heidy	22	22
Bella	22	22
Maurita	22	24
Milda	22	25
Breana	22	30
Karma	23	28
Lashandra	24	29
Teresa	25	25
Milford	25	26
Ryan	25	28
Parker	26	21
Hollis	26	23
Shirly	27	22
Sara	28	30
Bill	28	22
Antonetta	28	28
Pearline	29	25
Alisha	30	27
Ceola	30	26

copy 1

names	age	score
Parker	26	21
Heidy	22	22
Shirly	27	22
Arla	21	22
Bella	22	22
Bill	28	22
Hollis	26	23
Maurita	22	24
Milda	22	25
Pearline	29	25
Teresa	25	25
Ceola	30	26
Milford	25	26
Alisha	30	27
Antonetta	28	28
Ryan	25	28
Karma	23	28
Lashandra	24	29
Breana	22	30
Sara	28	30

copy 2

Why use a database?

1. More Structure
2. Sharing
3. Queries
4. Performance

Why not use a database?

It’s often overkill.

For many situations, a simple JSON or CSV is easier to use.

Outline

Tabular Data: CSVs vs. Databases

Common SQL Databases

Example: Madison bus-route data

SQL: Structured Query Language

Querying from Python

Popular SQL Databases

Microsoft® SQL Server®

ORACLE®

MySQL®

PostgreSQL

SQLite

There are minor differences in how you use these (e.g., what column types are available and how you query for data).

Most experience with one DB will translate to work with other DBs.

Popular SQL Databases

There are minor differences in how you use these (e.g., what column types are available and how you query for data).

Most experience with one DB will translate to work with other DBs.

in CS 301

Why learn SQLite?

- easy to install/use
- sqlite3 **module** comes with Python
- it’s public domain
- several billion deployments

Popular SQL Databases

Microsoft SQL Server

ORACLE

MySQL

PostgreSQL

SQLite

in CS 301

https://www.sqlite.org/mostdeployed.html

- Every Android device
- Every iPhone and iOS device
- Every Mac
- Every Windows 10 machine
- Every Firefox, Chrome, and Safari web browser
- Every instance of Skype
- Every instance of iTunes
- Every Dropbox client

Why learn SQLite?

- easy to install/use
- sqlite3 **module** comes with Python
- it's public domain
- several billion deployments

Outline

Tabular Data: CSVs vs. Databases

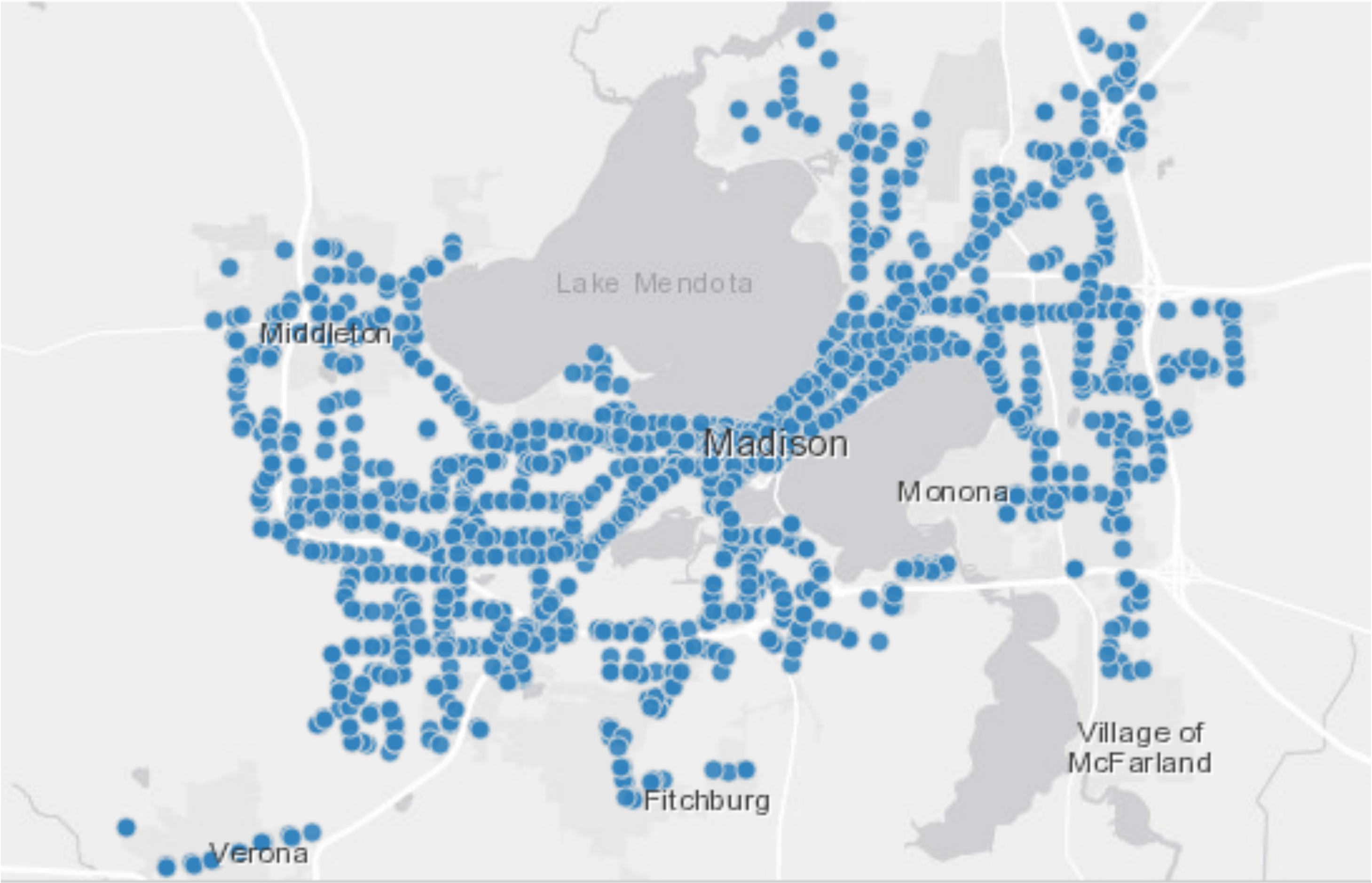
Common SQL Databases

Example: Madison bus-route data

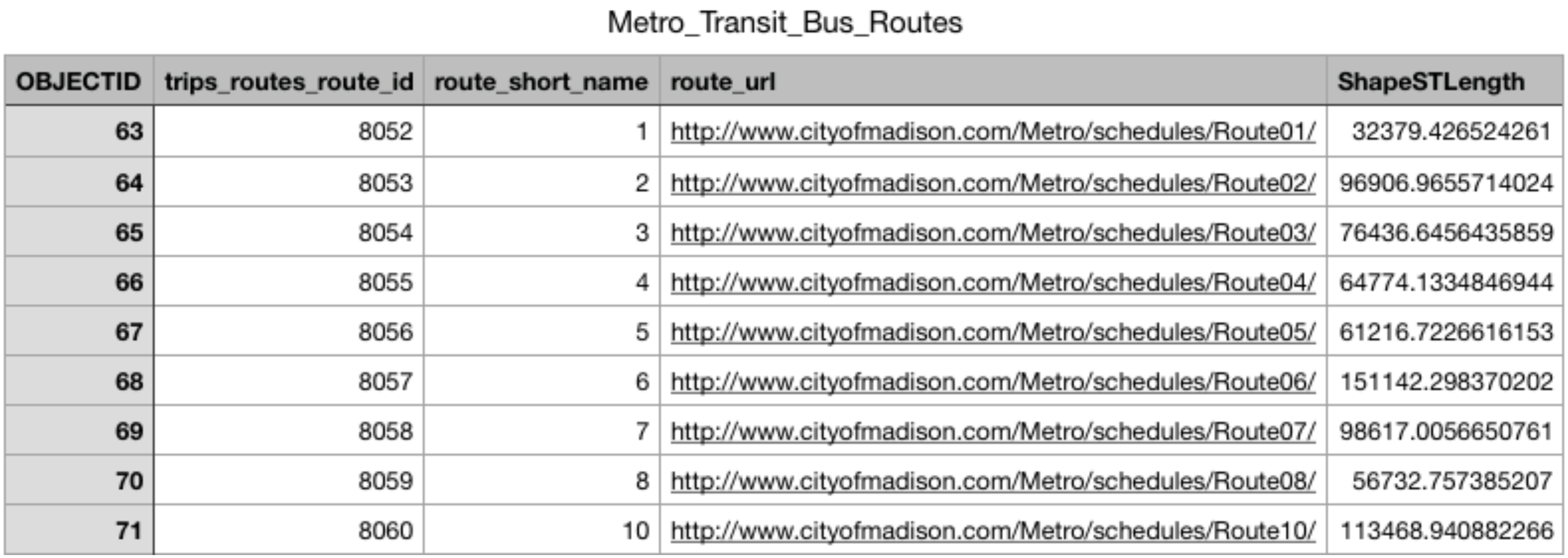
SQL: Structured Query Language

Querying from Python

Madison Bus Data: http://data-cityofmadison.opendata.arcgis.com/datasets/metro-transit-ridership-by-route-weekday

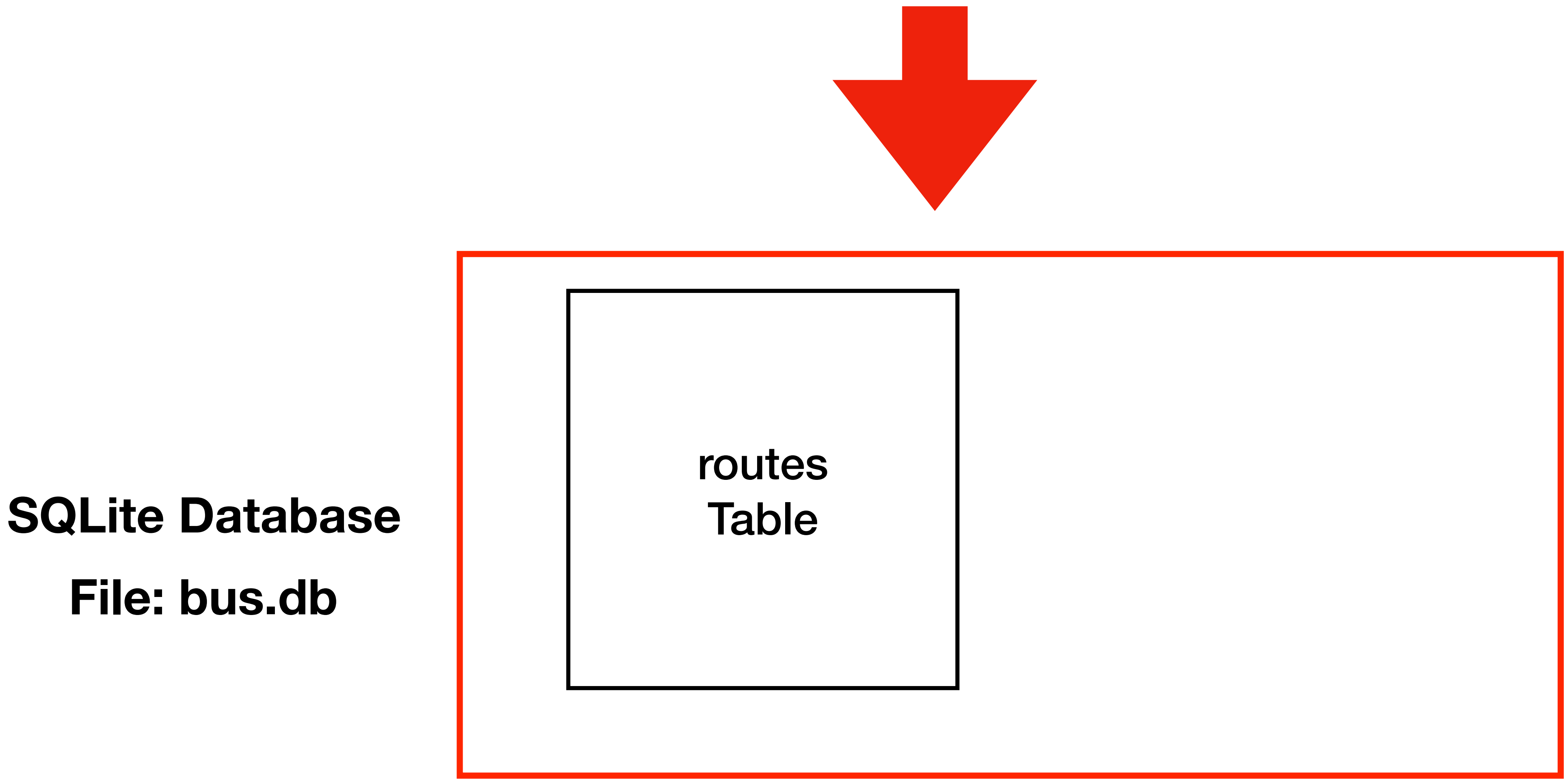


"Metro Transit ridership by route weekday. March, 2015. Caution should be used with this data. Daily bus stop boardings were estimated using a 12-day sample of weekday farebox records and AVL logs, and the GTFS file, from March 2015 from Metro Transit."



Metro_Transit_Bus_Routes

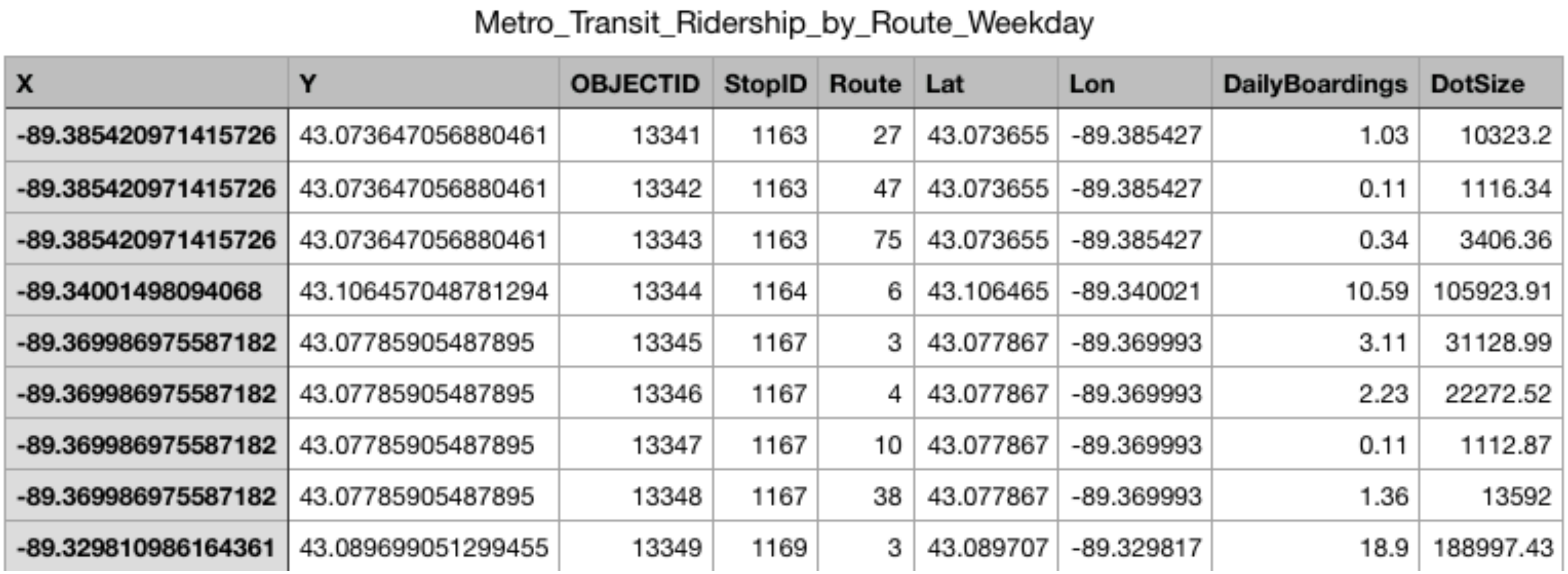
OBJECTID	trips_routes_route_id	route_short_name	route_url	ShapeSTLength
63	8052	1	http://www.cityofmadison.com/Metro/schedules/Route01/	32379.426524261
64	8053	2	http://www.cityofmadison.com/Metro/schedules/Route02/	96906.9655714024
65	8054	3	http://www.cityofmadison.com/Metro/schedules/Route03/	76436.6456435859
66	8055	4	http://www.cityofmadison.com/Metro/schedules/Route04/	64774.1334846944
67	8056	5	http://www.cityofmadison.com/Metro/schedules/Route05/	61216.7226616153
68	8057	6	http://www.cityofmadison.com/Metro/schedules/Route06/	151142.298370202
69	8058	7	http://www.cityofmadison.com/Metro/schedules/Route07/	98617.0056650761
70	8059	8	http://www.cityofmadison.com/Metro/schedules/Route08/	56732.757385207
71	8060	10	http://www.cityofmadison.com/Metro/schedules/Route10/	113468.940882266



SQLite Database

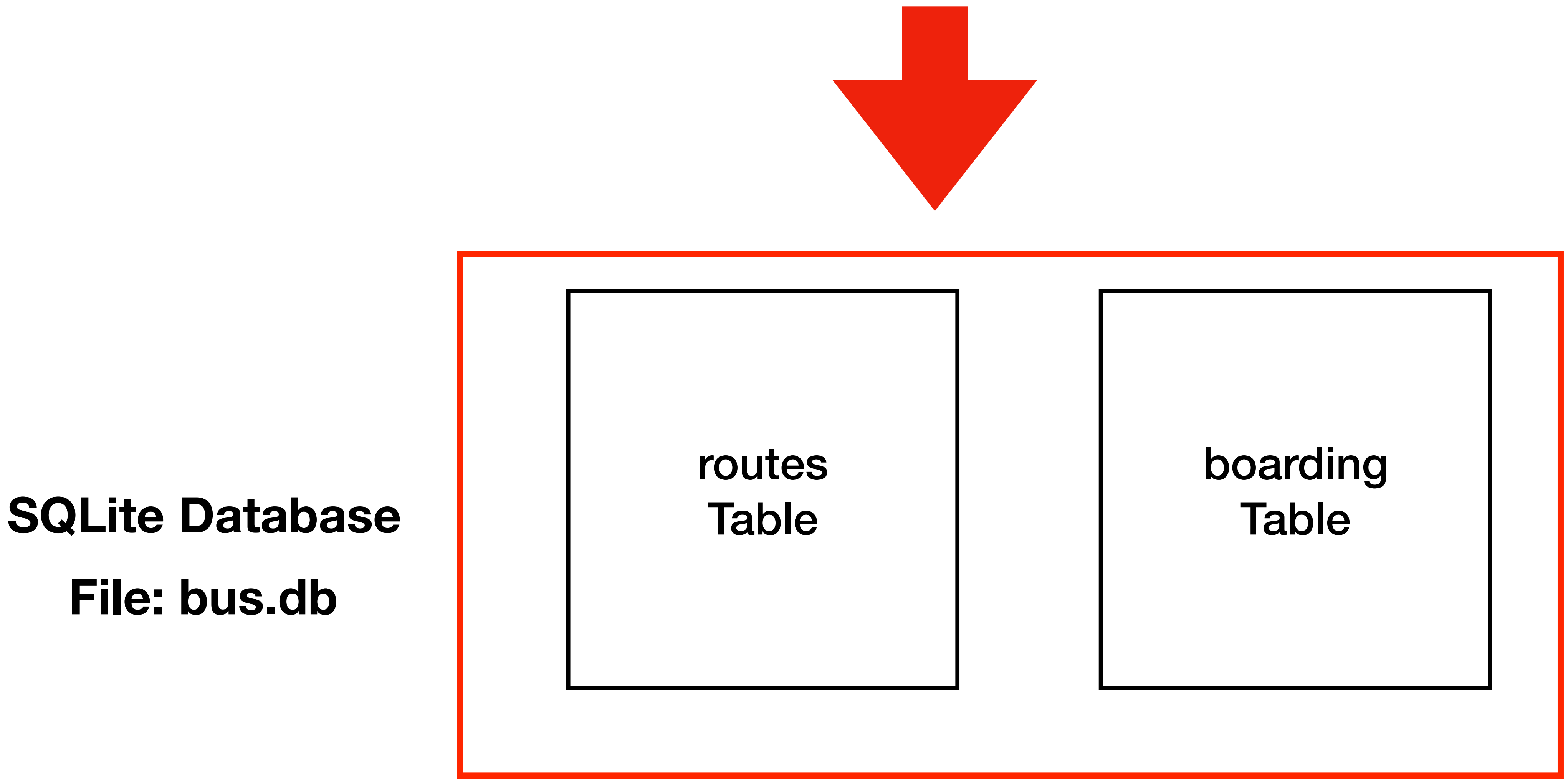
File: bus.db

routes
Table



Metro_Transit_Ridership_by_Route_Weekday

X	Y	OBJECTID	StopID	Route	Lat	Lon	DailyBoardings	DotSize
-89.385420971415726	43.073647056880461	13341	1163	27	43.073655	-89.385427	1.03	10323.2
-89.385420971415726	43.073647056880461	13342	1163	47	43.073655	-89.385427	0.11	1116.34
-89.385420971415726	43.073647056880461	13343	1163	75	43.073655	-89.385427	0.34	3406.36
-89.34001498094068	43.106457048781294	13344	1164	6	43.106465	-89.340021	10.59	105923.91
-89.369986975587182	43.07785905487895	13345	1167	3	43.077867	-89.369993	3.11	31128.99
-89.369986975587182	43.07785905487895	13346	1167	4	43.077867	-89.369993	2.23	22272.52
-89.369986975587182	43.07785905487895	13347	1167	10	43.077867	-89.369993	0.11	1112.87
-89.369986975587182	43.07785905487895	13348	1167	38	43.077867	-89.369993	1.36	13592
-89.329810986164361	43.089699051299455	13349	1169	3	43.089707	-89.329817	18.9	188997.43



SQLite Database

File: bus.db

routes Table

boarding Table

sqlite3 tool [DEMO]

Download bus.db:

https://github.com/tylerharter/caraza-harter-com/raw/master/tyler/cs301/fall18/materials/code/lec-31/bus.db

commands pasted for later review

```
ty-mac:lec-31$ sqlite3 bus.db
SQLite version 3.23.1 2018-04-10 17:39:29
Enter ".help" for usage hints.
sqlite> .tables
.tables
boarding  routes
sqlite> select * from routes;
select * from routes;
0|63|8052|1|http://www.cityofmadison.com/Metro/schedules/Route01/|32379
1|64|8053|2|http://www.cityofmadison.com/Metro/schedules/Route02/|96906
...
sqlite> select route_url from routes;
select route_url from routes;
http://www.cityofmadison.com/Metro/schedules/Route01/
http://www.cityofmadison.com/Metro/schedules/Route02/
http://www.cityofmadison.com/Metro/schedules/Route03/
...
sqlite> select * from boarding;
select * from boarding;
0|1163|27|43.073655|-89.385427|1.03
1|1163|47|43.073655|-89.385427|0.11
2|1163|75|43.073655|-89.385427|0.34
...
sqlite> select * from boarding limit 5;
select * from boarding limit 5;
0|1163|27|43.073655|-89.385427|1.03
1|1163|47|43.073655|-89.385427|0.11
2|1163|75|43.073655|-89.385427|0.34
3|1164|6|43.106465|-89.340021|10.59
4|1167|3|43.077867|-89.369993|3.11
sqlite> .schema
.schema
CREATE TABLE IF NOT EXISTS "boarding" (
...
```

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db

sqlite> a SQLite command
```

OR

```
sqlite> a SQL query
```

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db

sqlite> a SQLite command
```

OR

```
sqlite> a SQL query
```

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db

sqlite> .help
```

```
.archive ...           Manage SQL archives: ".archive --help" for details
.auth ON|OFF           Show authorizer callbacks
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail on|off           Stop after hitting an error.  Default OFF
.binary on|off         Turn binary output on or off.  Default OFF
.cd DIRECTORY          Change the working directory to DIRECTORY
...
```

get list of SQLite commands
(all begin with a period)

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db

sqlite> .tables
boarding   routes
```

print list of tables in the database

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db

sqlite> .schema
```

```
CREATE TABLE IF NOT EXISTS "boarding" (
"index" INTEGER,
  "StopID" INTEGER,
  "Route" INTEGER,
  "Lat" REAL,
  "Lon" REAL,
  "DailyBoardings" REAL
);
CREATE INDEX "ix_boarding_index"ON "boarding" ("index");
CREATE TABLE IF NOT EXISTS "routes" (
"index" INTEGER,
  "OBJECTID" INTEGER,
  "trips_routes_route_id" INTEGER,
  "route_short_name" INTEGER,
  "route_url" TEXT,
  "ShapeSTLength" REAL
);
CREATE INDEX "ix_routes_index"ON "routes" ("index");
```

prints SQL code to create those tables

```
CREATE TABLE IF NOT EXISTS "boarding" (
"index" INTEGER,
  "StopID" INTEGER,
  "Route" INTEGER,
  "Lat" REAL,
  "Lon" REAL,
  "DailyBoardings" REAL
);
CREATE INDEX "ix_boarding_index"ON "boarding" ("index");
CREATE TABLE IF NOT EXISTS "routes" (
"index" INTEGER,
  "OBJECTID" INTEGER,
  "trips_routes_route_id" INTEGER,
  "route_short_name" INTEGER,
  "route_url" TEXT,
  "ShapeSTLength" REAL
);
CREATE INDEX "ix_routes_index"ON "routes" ("index");
CREATE TABLE IF NOT EXISTS "boarding" (
"index" INTEGER,
  "StopID" INTEGER,
  "Route" INTEGER,
  "Lat" REAL,
  "Lon" REAL,
  "DailyBoardings" REAL
);
CREATE INDEX "ix_boarding_index"ON "boarding" ("index");
CREATE TABLE IF NOT EXISTS "routes" (
"index" INTEGER,
  "OBJECTID" INTEGER,
  "trips_routes_route_id" INTEGER,
  "route_short_name" INTEGER,
  "route_url" TEXT,
  "ShapeSTLength" REAL
);
CREATE INDEX "ix_routes_index"ON "routes" ("index");
```

table names

```
CREATE TABLE IF NOT EXISTS "boarding" (
"index" INTEGER,
  "StopID" INTEGER,
  "Route" INTEGER,
  "Lat" REAL,
  "Lon" REAL,
  "DailyBoardings" REAL
);
CREATE INDEX "ix_boarding_index"ON "boarding" ("index");
CREATE TABLE IF NOT EXISTS "routes" (
"index" INTEGER,
  "OBJECTID" INTEGER,
  "trips_routes_route_id" INTEGER,
  "route_short_name" INTEGER,
  "route_url" TEXT,
  "ShapeSTLength" REAL
);
CREATE INDEX "ix_routes_index"ON "routes" ("index");
```

look for column names in parens

columns

- index
- StopID
- Route
- Lat
- Lon
- Daily Boardings

```
CREATE TABLE IF NOT EXISTS "boarding" (
"index" INTEGER,
  "StopID" INTEGER,
  "Route" INTEGER,
  "Lat" REAL,
  "Lon" REAL,
  "DailyBoardings" REAL
);
CREATE INDEX "ix_boarding_index"ON "boarding" ("index");
CREATE TABLE IF NOT EXISTS "routes" (
"index" INTEGER,
  "OBJECTID" INTEGER,
  "trips_routes_route_id" INTEGER,
  "route_short_name" INTEGER,
  "route_url" TEXT,
  "ShapeSTLength" REAL
);
CREATE INDEX "ix_routes_index"ON "routes" ("index");
```

types...

```
CREATE TABLE IF NOT EXISTS "boarding" (
"index" INTEGER,
  "StopID" INTEGER,
  "Route" INTEGER,
  "Lat" REAL,
  "Lon" REAL,
  "DailyBoardings" REAL
);
CREATE INDEX "ix_boarding_index"ON "boarding" ("index");
CREATE TABLE IF NOT EXISTS "routes" (
"index" INTEGER,
  "OBJECTID" INTEGER,
  "trips_routes_route_id" INTEGER,
  "route_short_name" INTEGER,
  "route_url" TEXT,
  "ShapeSTLength" REAL
);
CREATE INDEX "ix_routes_index"ON "routes" ("index");
```

Note: we're glossing over lots of details because we'll use pandas to create these columns in table **boarding**:

- index
- StopID
- Route
- Lat
- Lon
- DailyBoardings

columns in table **routes**:

- index
- OBJECTID
- trips_routes_route_id,
- route_short_name
- route_url
- ShapeSTLength

columns in table **boarding**:

- index
- StopID
- Route
- Lat
- Lon
- DailyBoardings

columns in table **routes**:

- index
- OBJECTID
- trips_routes_route_id,
- route_short_name
- route_url
- ShapeSTLength

Once we identify the table and column names, how can ask questions?

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db
```

```
sqlite> a SQLite command (starts with ".")
```

OR

```
sqlite> a SQL query;
```

sqlite3 prompt

```
ty-mac:lec-31$ sqlite3 bus.db
```

```
sqlite> a SQLite command (starts with ".")
```

OR

```
sqlite> a SQL query;
```

Outline

Tabular Data: CSVs vs. Databases

Common SQL Databases

Example: Madison bus-route data

SQL: Structured Query Language

Querying from Python

Overview: Narrowing Down

table 1

table 2

table 3

Overview: Narrowing Down

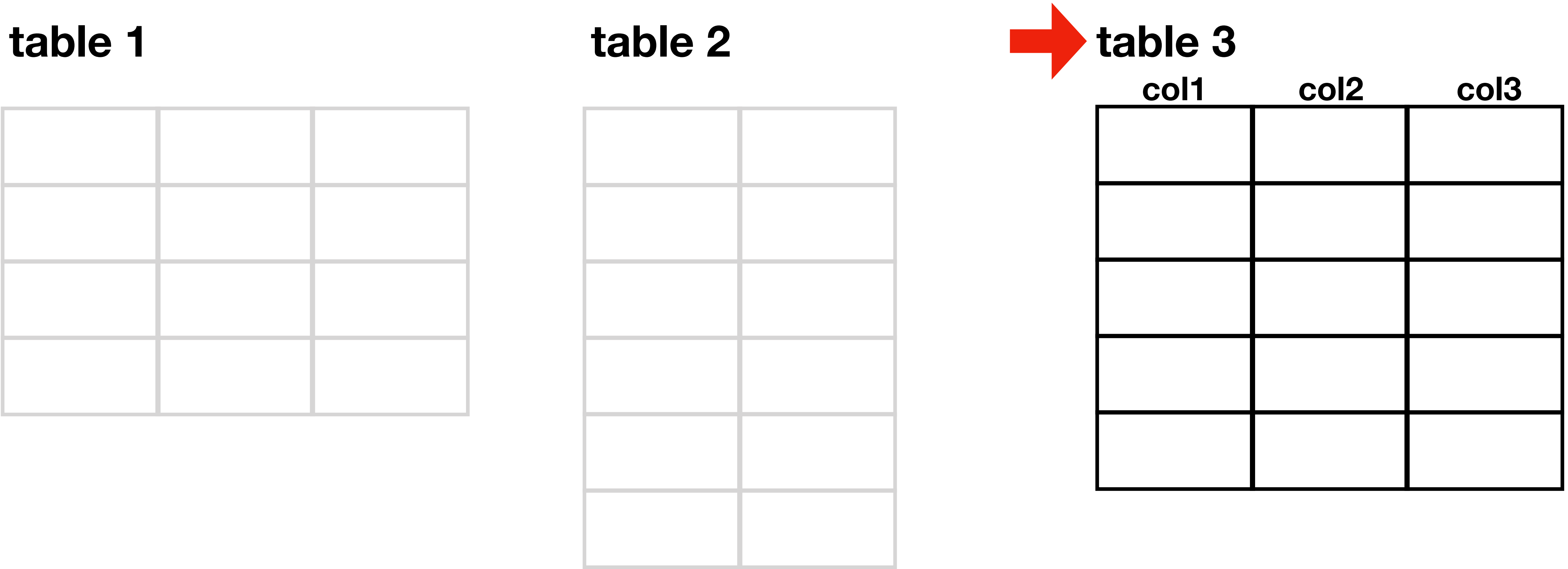


table 1

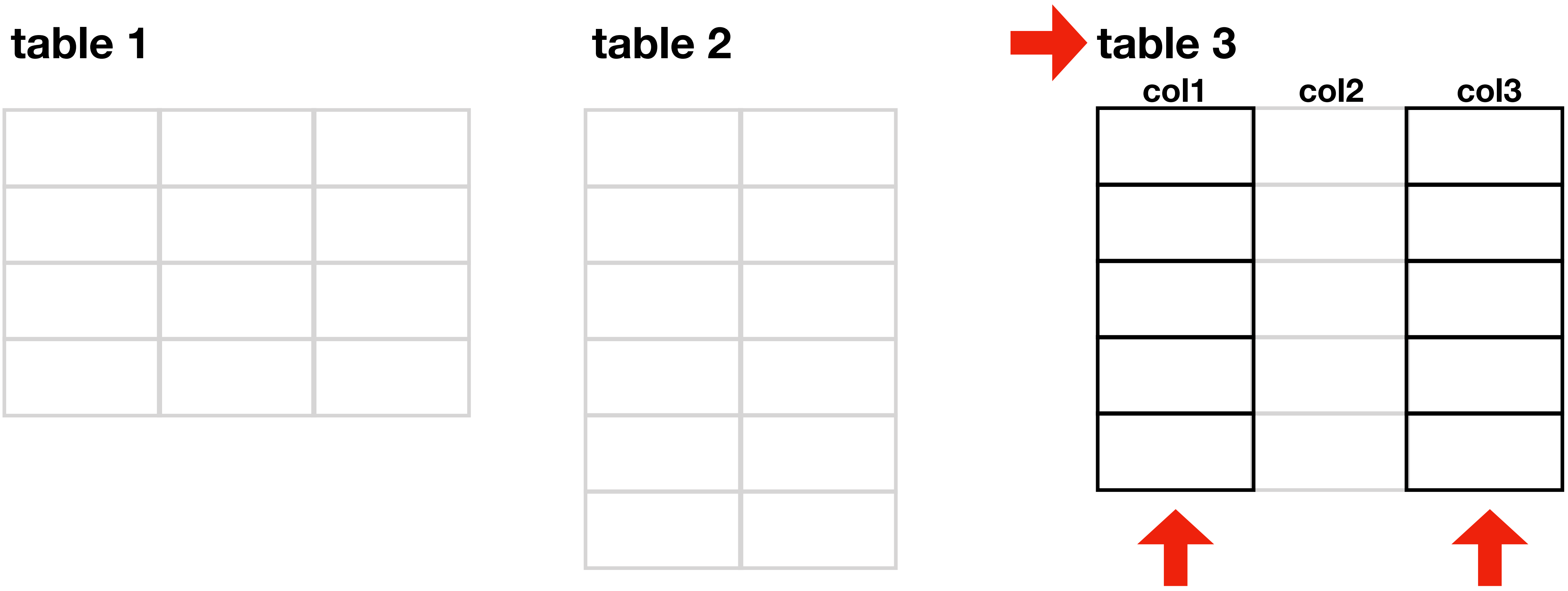
table 2

table 3

col1 col2 col3

FROM: which table?

Overview: Narrowing Down



FROM: which table?
SELECT: which columns?

Overview: Narrowing Down

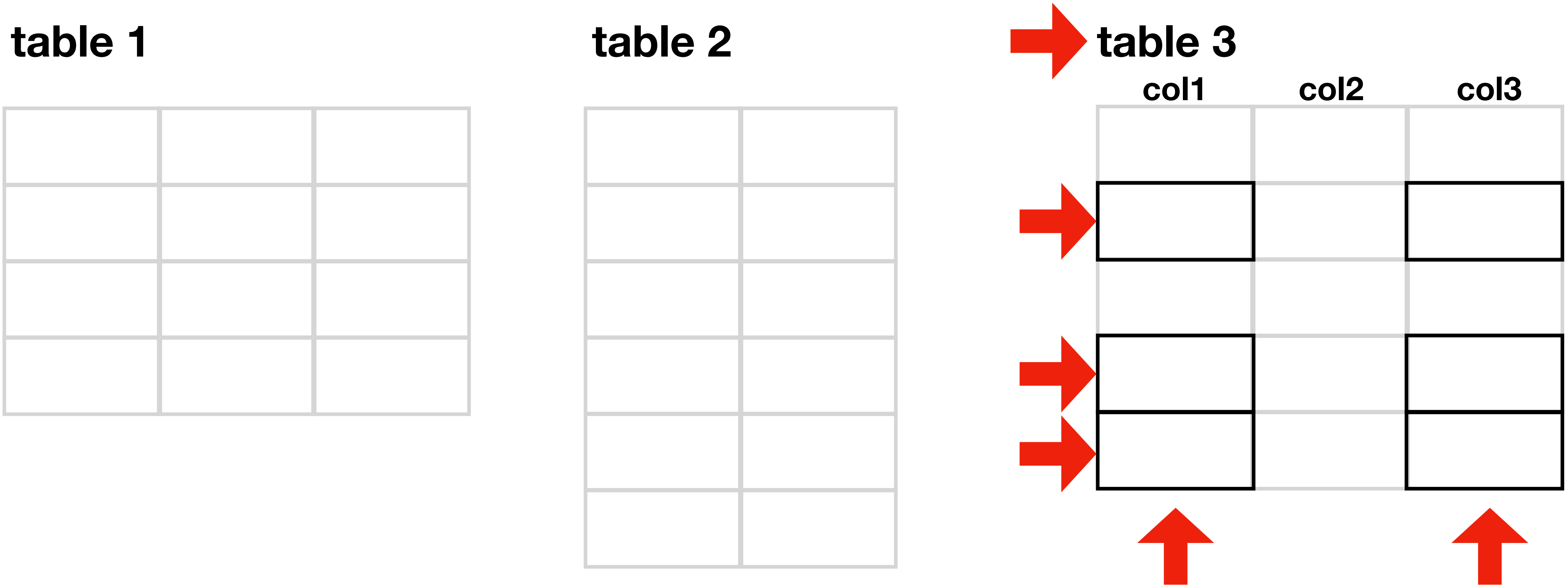


table 1

table 2

table 3

col1 col2 col3

FROM: which table?
SELECT: which columns?
WHERE: which rows?

Overview: Narrowing Down

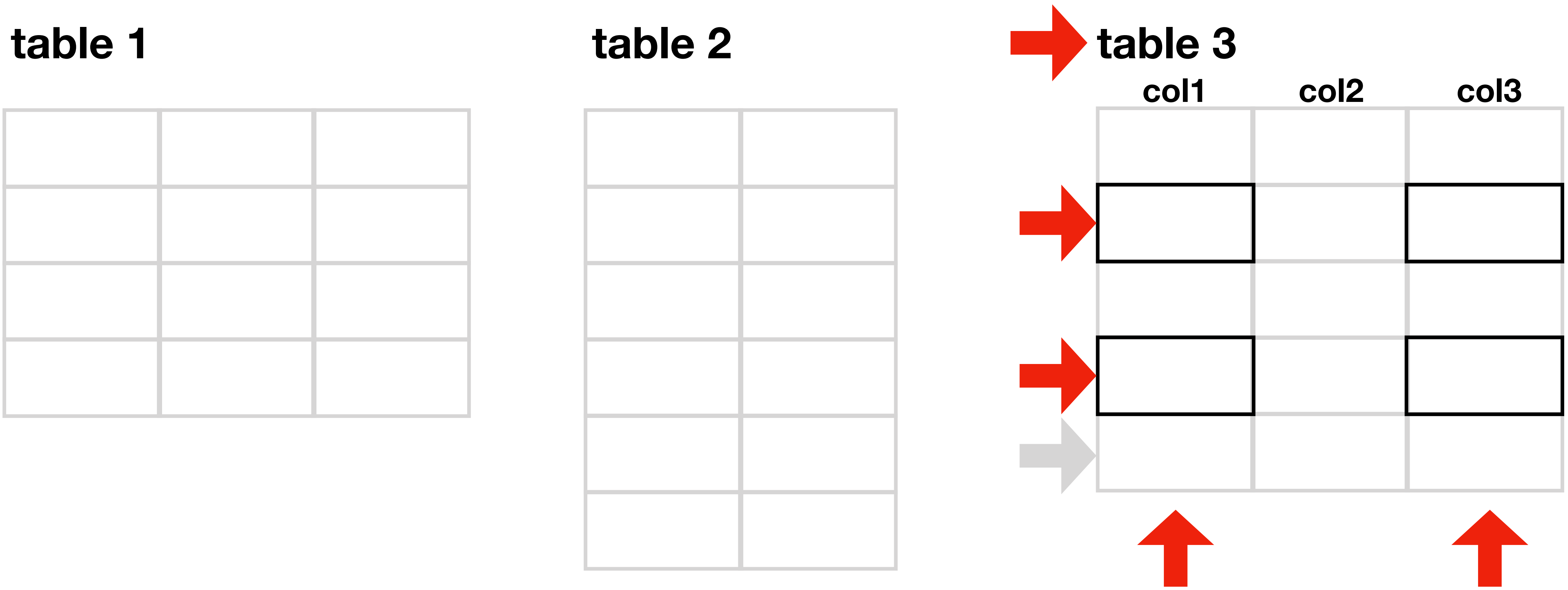


table 1

table 2

table 3

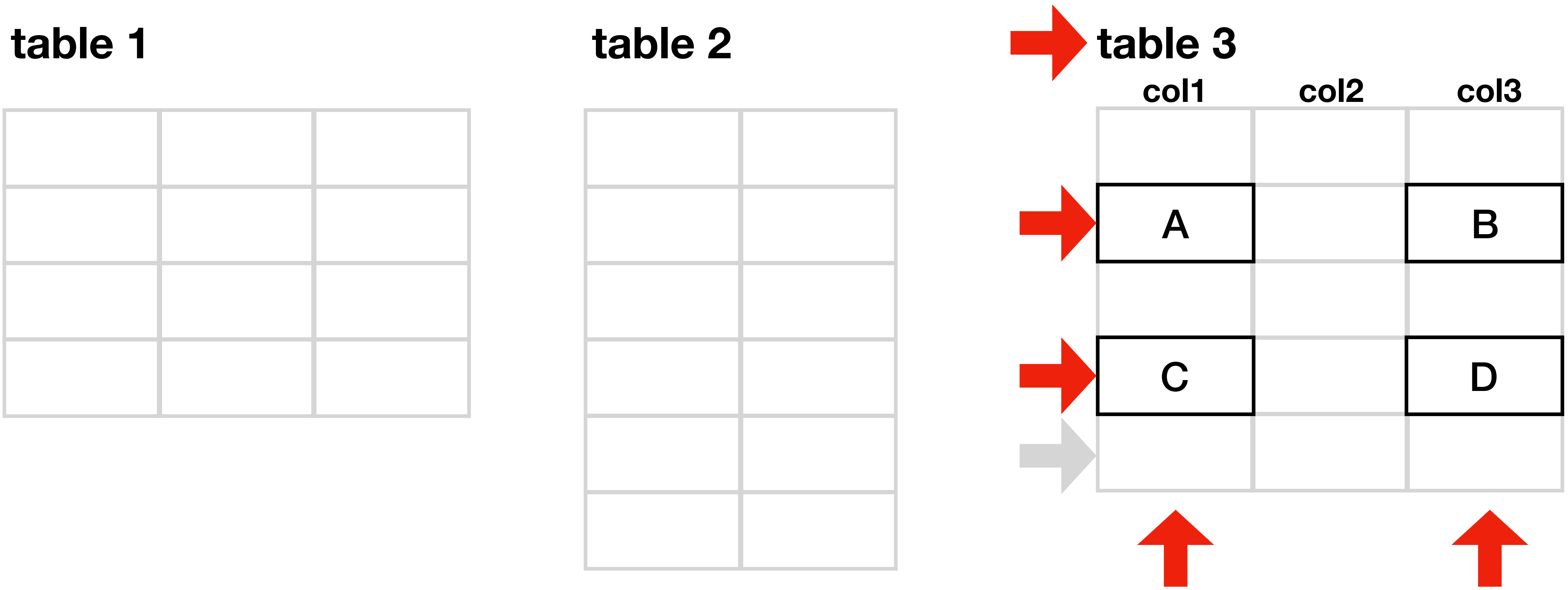
col1

col2

col3

FROM: which table?
SELECT: which columns?
WHERE: which rows?
LIMIT: how many rows?

Overview: Narrowing Down



FROM: which table?
SELECT: which columns?
WHERE: which rows?
LIMIT: how many rows?

a query result looks like a table

col1	col3
A	B
C	D

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

SELECT □ FROM □ ;

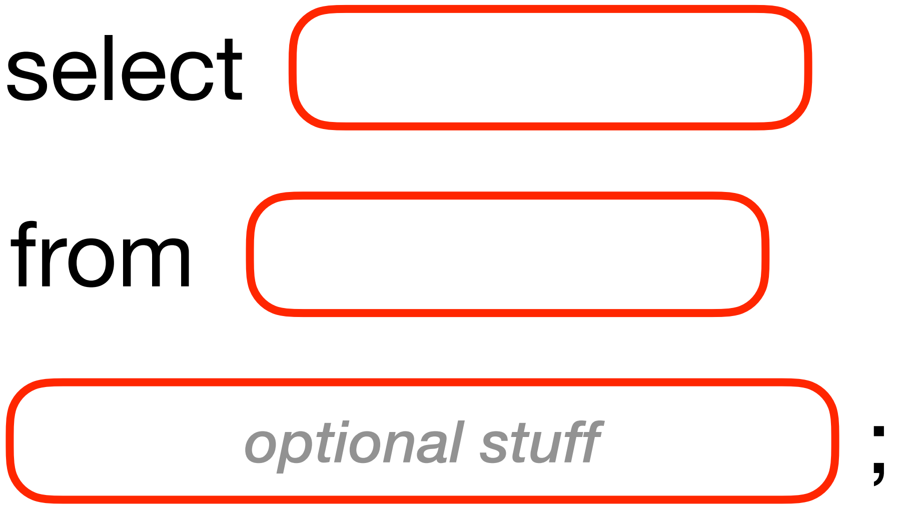
SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select □
from □ ;
```

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):



select

from

optional stuff ;

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

select

from table name ;

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select
from boarding;
```

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select  which columns

from boarding;
```

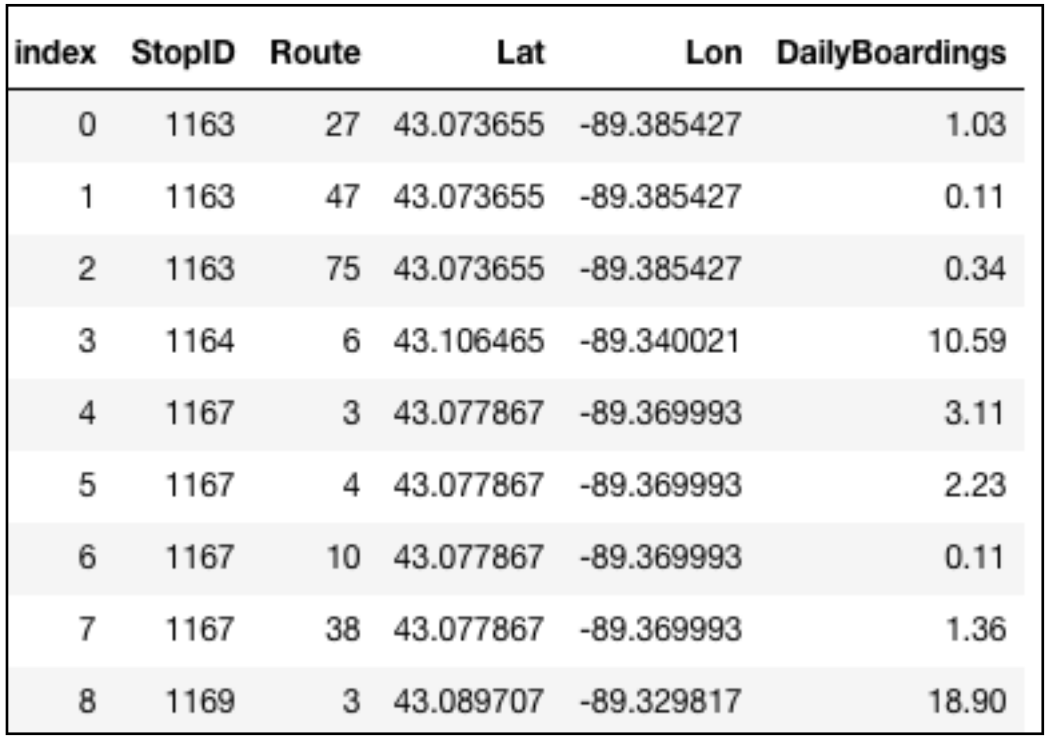
SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

star means all of them

```
select *
from boarding;
```

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
0	1163	27	43.073655	-89.385427	1.03
1	1163	47	43.073655	-89.385427	0.11
2	1163	75	43.073655	-89.385427	0.34
3	1164	6	43.106465	-89.340021	10.59
4	1167	3	43.077867	-89.369993	3.11
5	1167	4	43.077867	-89.369993	2.23
6	1167	10	43.077867	-89.369993	0.11
7	1167	38	43.077867	-89.369993	1.36
8	1169	3	43.089707	-89.329817	18.90

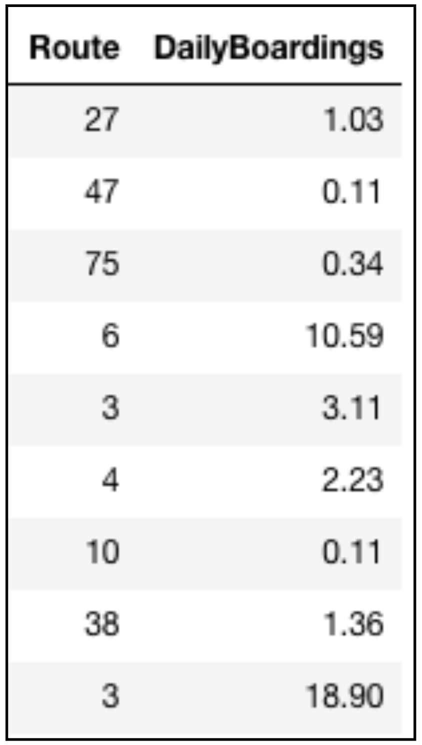
...

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select Route, DailyBoardings
from boarding;
```

Result:



Route	DailyBoardings
27	1.03
47	0.11
75	0.34
6	10.59
3	3.11
4	2.23
10	0.11
38	1.36
3	18.90

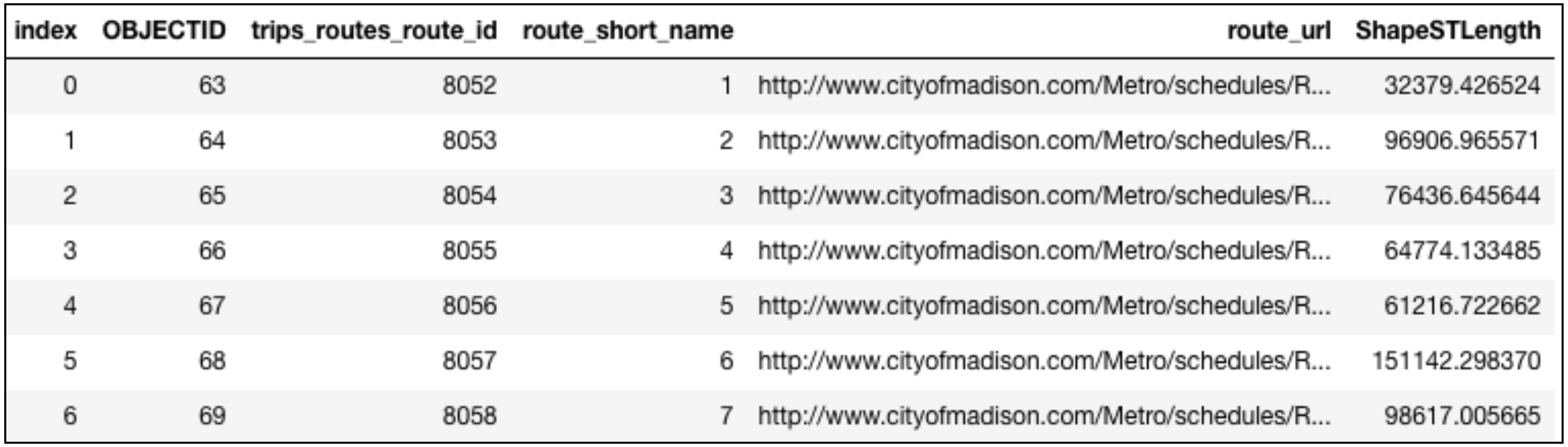
...

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select *
from routes;
```

Result:



index	OBJECTID	trips_routes_route_id	route_short_name	route_url	ShapeSTLength
0	63	8052	1	http://www.cityofmadison.com/Metro/schedules/R...	32379.426524
1	64	8053	2	http://www.cityofmadison.com/Metro/schedules/R...	96906.965571
2	65	8054	3	http://www.cityofmadison.com/Metro/schedules/R...	76436.645644
3	66	8055	4	http://www.cityofmadison.com/Metro/schedules/R...	64774.133485
4	67	8056	5	http://www.cityofmadison.com/Metro/schedules/R...	61216.722662
5	68	8057	6	http://www.cityofmadison.com/Metro/schedules/R...	151142.298370
6	69	8058	7	http://www.cityofmadison.com/Metro/schedules/R...	98617.005665

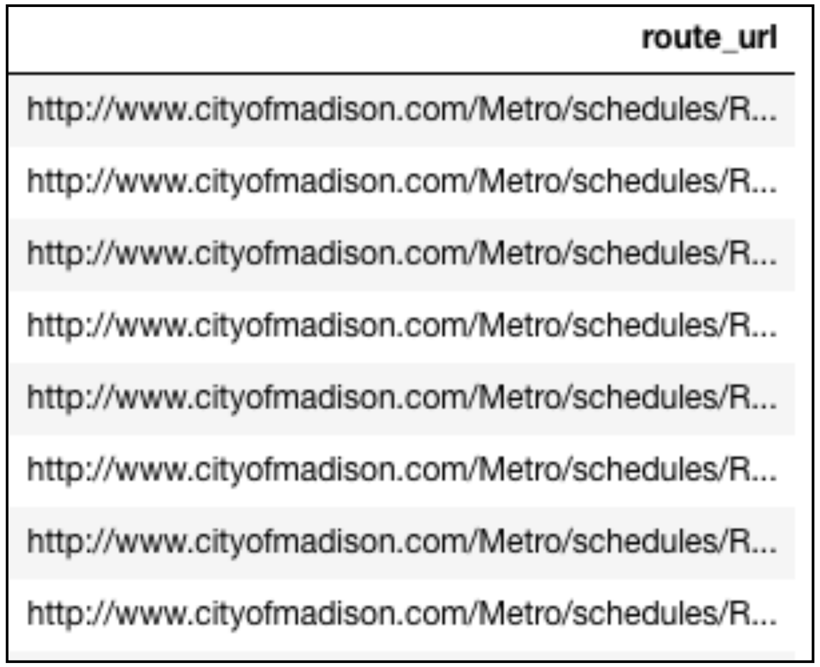
...

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select route_url
from routes;
```

Result:

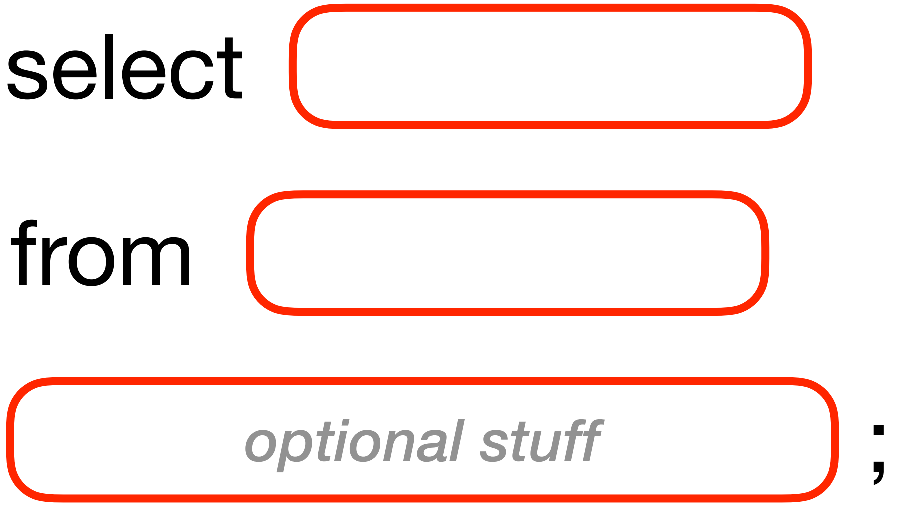


route_url
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...
http://www.cityofmadison.com/Metro/schedules/R...

…

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):



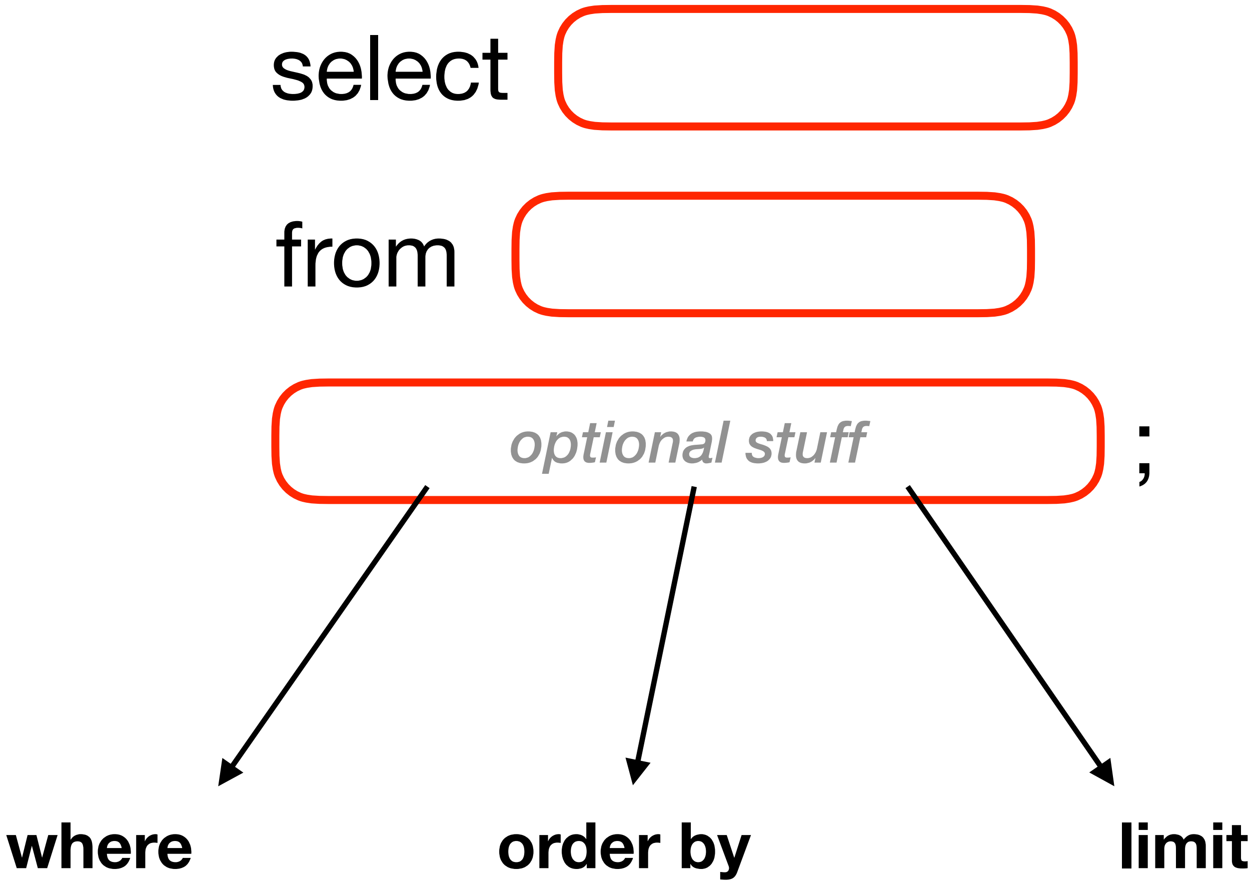
select

from

optional stuff ;

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):



select

from

optional stuff ;

where

order by

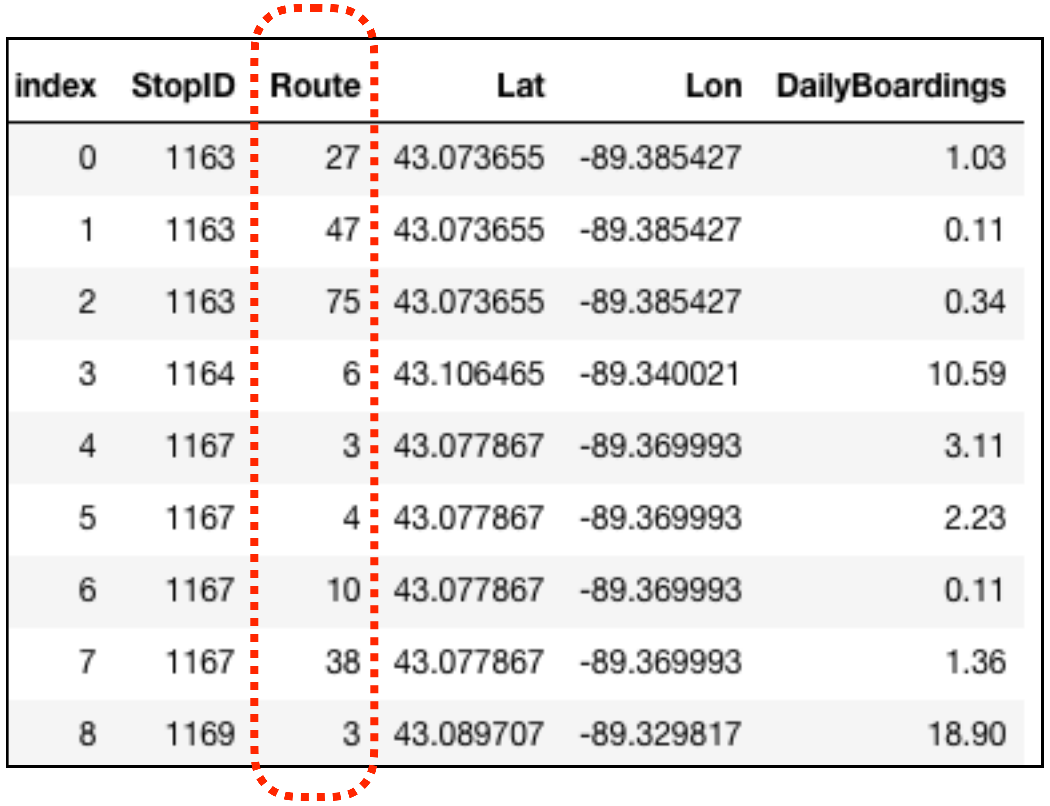
limit

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select *
from boarding;
```

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
0	1163	27	43.073655	-89.385427	1.03
1	1163	47	43.073655	-89.385427	0.11
2	1163	75	43.073655	-89.385427	0.34
3	1164	6	43.106465	-89.340021	10.59
4	1167	3	43.077867	-89.369993	3.11
5	1167	4	43.077867	-89.369993	2.23
6	1167	10	43.077867	-89.369993	0.11
7	1167	38	43.077867	-89.369993	1.36
8	1169	3	43.089707	-89.329817	18.90

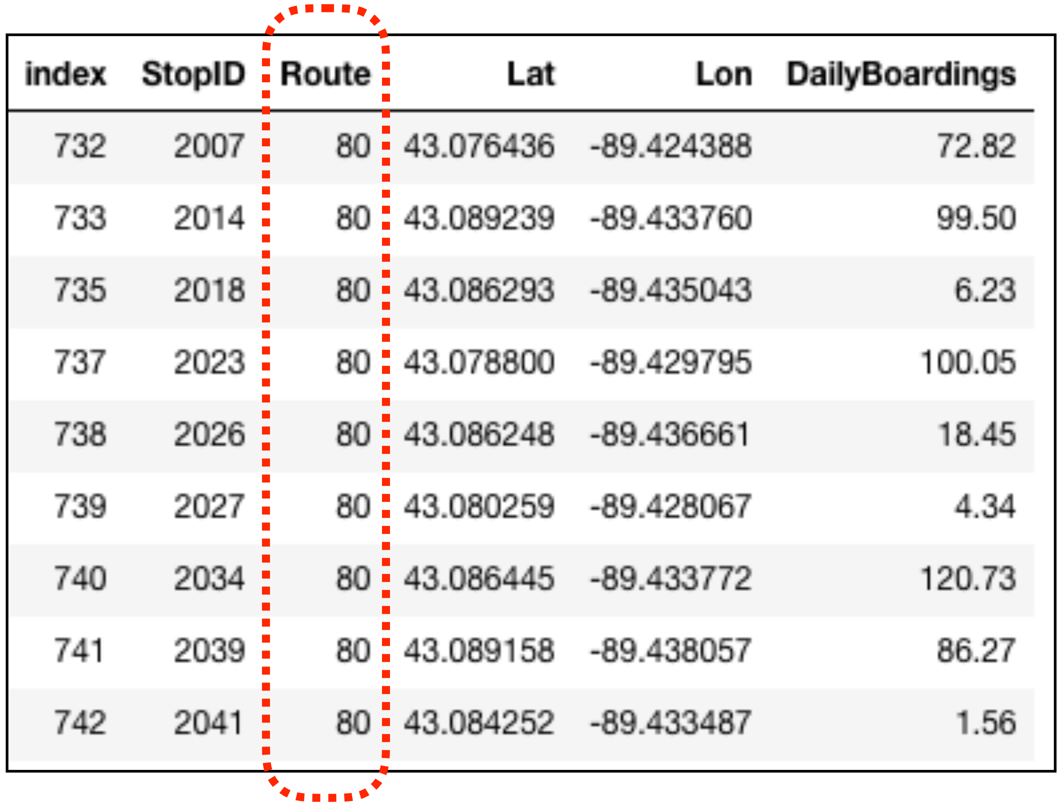
...

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select *
from boarding
where Route = 80;
```

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
732	2007	80	43.076436	-89.424388	72.82
733	2014	80	43.089239	-89.433760	99.50
735	2018	80	43.086293	-89.435043	6.23
737	2023	80	43.078800	-89.429795	100.05
738	2026	80	43.086248	-89.436661	18.45
739	2027	80	43.080259	-89.428067	4.34
740	2034	80	43.086445	-89.433772	120.73
741	2039	80	43.089158	-89.438057	86.27
742	2041	80	43.084252	-89.433487	1.56

...

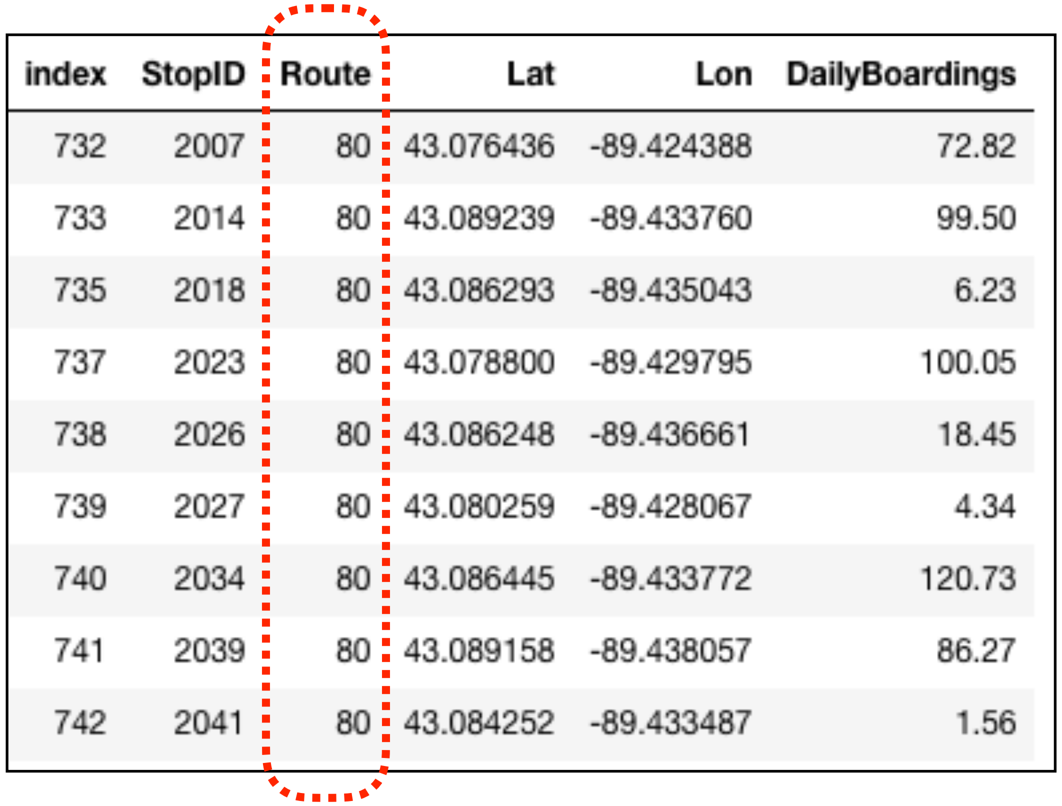
SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select *
from boarding
where Route = 80;
```

note SQL only has one equal sign for equality!

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
732	2007	80	43.076436	-89.424388	72.82
733	2014	80	43.089239	-89.433760	99.50
735	2018	80	43.086293	-89.435043	6.23
737	2023	80	43.078800	-89.429795	100.05
738	2026	80	43.086248	-89.436661	18.45
739	2027	80	43.080259	-89.428067	4.34
740	2034	80	43.086445	-89.433772	120.73
741	2039	80	43.089158	-89.438057	86.27
742	2041	80	43.084252	-89.433487	1.56

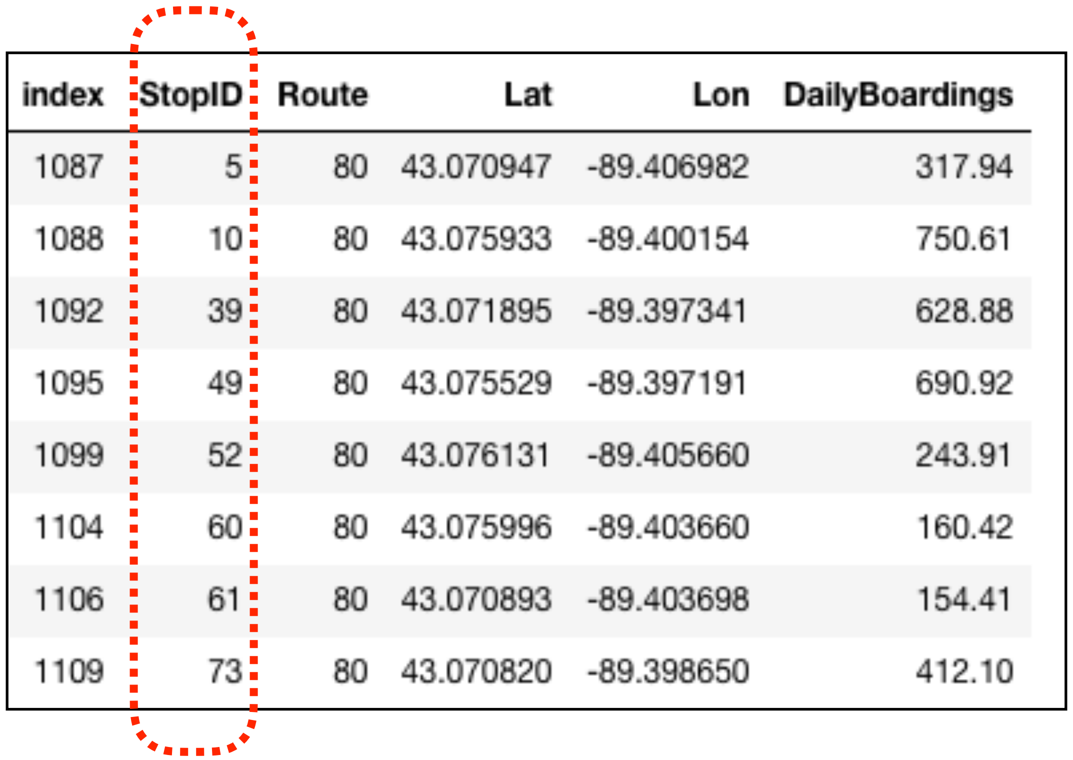
...

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select *
from boarding
where Route = 80
order by StopID;
```

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
1087	5	80	43.070947	-89.406982	317.94
1088	10	80	43.075933	-89.400154	750.61
1092	39	80	43.071895	-89.397341	628.88
1095	49	80	43.075529	-89.397191	690.92
1099	52	80	43.076131	-89.405660	243.91
1104	60	80	43.075996	-89.403660	160.42
1106	61	80	43.070893	-89.403698	154.41
1109	73	80	43.070820	-89.398650	412.10

...

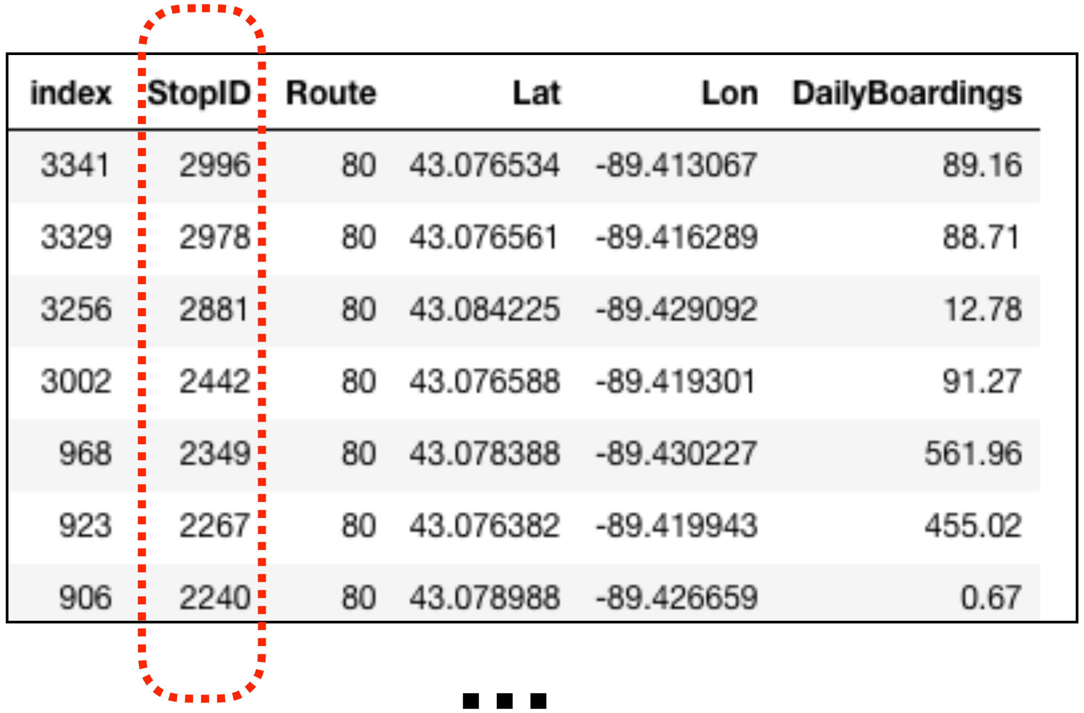
SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select *
from boarding
where Route = 80
order by StopID DESC;
```

descending means biggest first

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
3341	2996	80	43.076534	-89.413067	89.16
3329	2978	80	43.076561	-89.416289	88.71
3256	2881	80	43.084225	-89.429092	12.78
3002	2442	80	43.076588	-89.419301	91.27
968	2349	80	43.078388	-89.430227	561.96
923	2267	80	43.076382	-89.419943	455.02
906	2240	80	43.078988	-89.426659	0.67

...

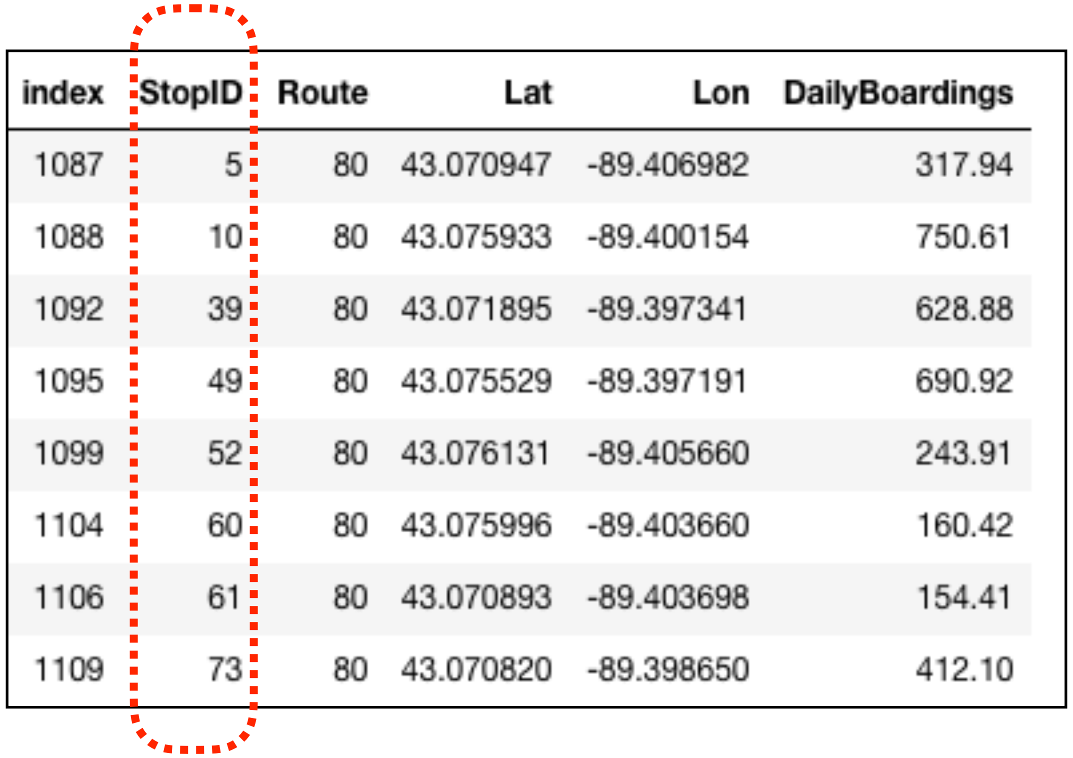
SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select *
from boarding
where Route = 80
order by StopID ASC;
```

ascending means
biggest first

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
1087	5	80	43.070947	-89.406982	317.94
1088	10	80	43.075933	-89.400154	750.61
1092	39	80	43.071895	-89.397341	628.88
1095	49	80	43.075529	-89.397191	690.92
1099	52	80	43.076131	-89.405660	243.91
1104	60	80	43.075996	-89.403660	160.42
1106	61	80	43.070893	-89.403698	154.41
1109	73	80	43.070820	-89.398650	412.10

...

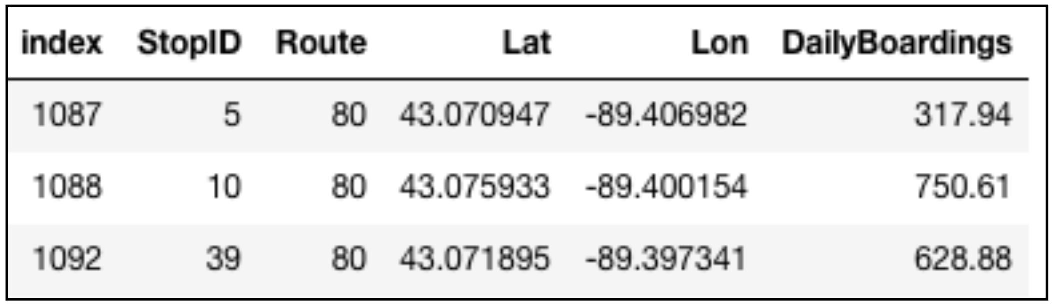
SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select *
from boarding
where Route = 80
order by StopID ASC
limit 3;
```

only show the top N results

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
1087	5	80	43.070947	-89.406982	317.94
1088	10	80	43.075933	-89.400154	750.61
1092	39	80	43.071895	-89.397341	628.88

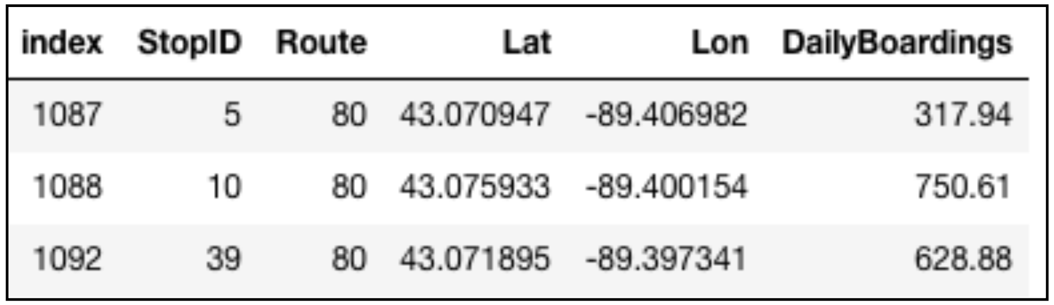
3 results

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don’t matter):

```
select *
from boarding
where Route = 80
order by StopID ASC
limit 3;
```

Result:



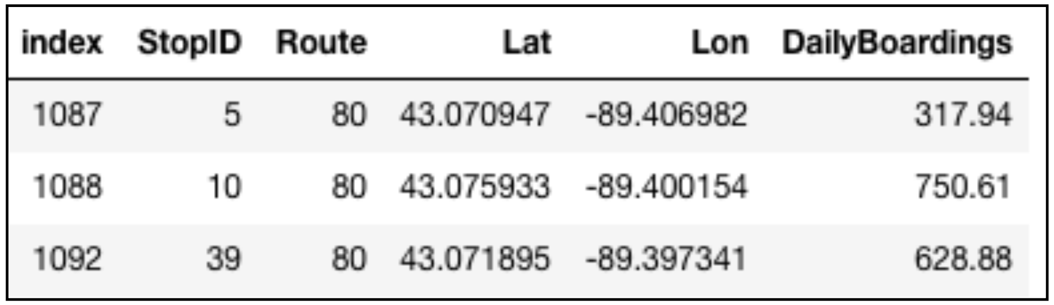
index	StopID	Route	Lat	Lon	DailyBoardings
1087	5	80	43.070947	-89.406982	317.94
1088	10	80	43.075933	-89.400154	750.61
1092	39	80	43.071895	-89.397341	628.88

SQL Queries: How to ask a DB questions

Syntax for SELECT (case and spacing don't matter):

```
select *
from boarding
where Route = 80
order by StopID ASC
limit 3;
```

Result:



index	StopID	Route	Lat	Lon	DailyBoardings
1087	5	80	43.070947	-89.406982	317.94
1088	10	80	43.075933	-89.400154	750.61
1092	39	80	43.071895	-89.397341	628.88

You can use any combination of where, order by, and limit.
But whichever you use, they must appear in that order!

Outline

Tabular Data: CSVs vs. Databases

Common SQL Databases

Example: Madison bus-route data

SQL: Structured Query Language

Querying from Python

Modules we’ve learned this semester

- math
- collections
- json
- csv
- sys
- os
- copy
- recordclass
- requests
- bs4 (BeautifulSoup)
- pandas ← integrates with SQLite
- sqlite3 ← directly access SQLite databases (comes with Python)

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")
```

database filename

- represented as a string
- will create if doesn’t already exist

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")
```

connect for databases is analogous to **open** for files

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")
```

a **connection object** for
databases is analogous to
file object for files

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")
```

close it at the end

```
conn.close()
```

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")

results = conn.execute(query_string)
```

instead of reading (as to a file), we **execute** SQL queries on the database

```
conn.close()
```

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")

results = conn.execute("select * from boarding")
```

the **SQL query** is just a string
that is passed to the **execute**

```
conn.close()
```

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")

results = conn.execute("select * from boarding")
```

results is an **iterator** over **tuples**
we get by running the query

```
conn.close()
```

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")

results = conn.execute("select * from boarding")

for row in results:
    print(row)


conn.close()
```

Output:

```
(0, 1163, 27, 43.073655, -89.385427, 1.03)
(1, 1163, 47, 43.073655, -89.385427, 0.11)
(2, 1163, 75, 43.073655, -89.385427, 0.34)
(3, 1164, 6, 43.106465, -89.340021, 10.59)
(4, 1167, 3, 43.077867, -89.36999300000001, 3.11)
(5, 1167, 4, 43.077867, -89.36999300000001, 2.23)
...
```

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")

results = conn.execute("select * from boarding")

for row in results:
    print(row)


conn.close()
```

```
select * from sqlite_master
```

(sqlite_master describes other tables)

Output:

```
(0, 1163, 27, 43.073655, -89.385427, 1.03)
(1, 1163, 47, 43.073655, -89.385427, 0.11)
(2, 1163, 75, 43.073655, -89.385427, 0.34)
(3, 1164, 6, 43.106465, -89.340021, 10.59)
(4, 1167, 3, 43.077867, -89.36999300000001, 3.11)
(5, 1167, 4, 43.077867, -89.36999300000001, 2.23)
...
```

sqlite3

```
import sqlite3

conn = sqlite3.connect("file.db")

results = conn.execute("select * from boarding")

for row in results:
    print(row)


conn.close()
```

**Instead of looping over tuples,
Pandas can execute the query
and give a DataFrame of results**

sqlite3

```
import sqlite3
import pandas as pd
conn = sqlite3.connect("file.db")

results = conn.execute("select * from boarding")

for row in results:
    print(row)


conn.close()
```

Instead of looping over tuples, Pandas can execute the query and give a DataFrame of results

sqlite3

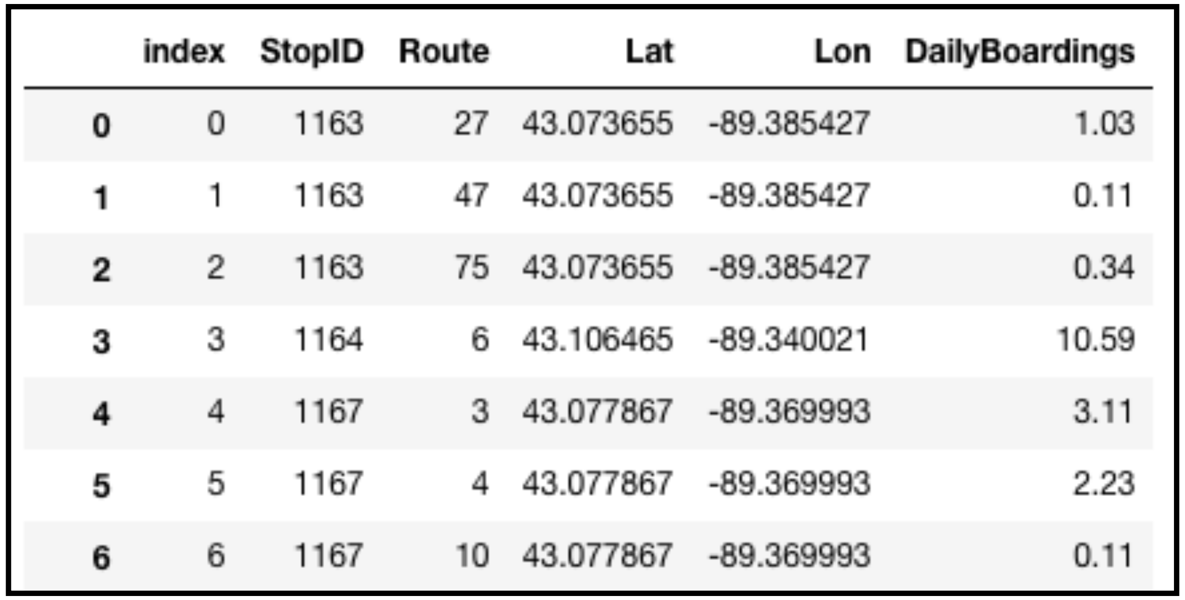
```
import sqlite3
import pandas as pd
conn = sqlite3.connect("file.db")

df = pd.read_sql("select * from boarding", conn)
```

run this **query**

on this **database**

```
conn.close()
```

df:



	index	StopID	Route	Lat	Lon	DailyBoardings
0	0	1163	27	43.073655	-89.385427	1.03
1	1	1163	47	43.073655	-89.385427	0.11
2	2	1163	75	43.073655	-89.385427	0.34
3	3	1164	6	43.106465	-89.340021	10.59
4	4	1167	3	43.077867	-89.369993	3.11
5	5	1167	4	43.077867	-89.369993	2.23
6	6	1167	10	43.077867	-89.369993	0.11

...

Demo 1: How Many People Ride the Bus

Goal: add up all boardings across all bus stops/routes

Input:

- bus.db
- use `DailyBoardings` column in `boarding` table
- **Challenge**: use `conn.execute` call (instead of pandas)

Output:

- total riders

Demo 2: West-most Bus Route

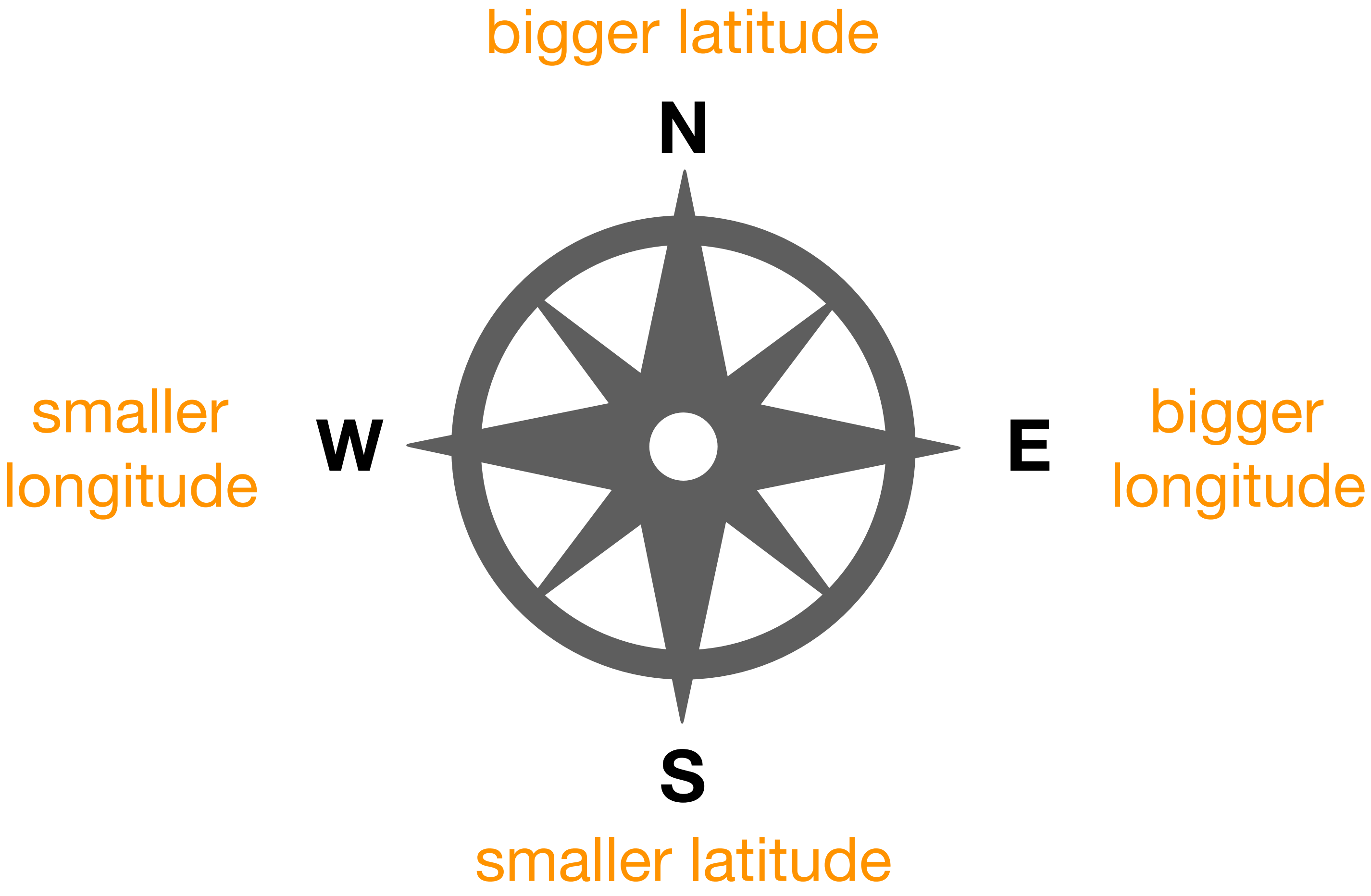
Goal: which Madison bus goes farthest west?

Input:

- bus.db

Output:

- route number of bus that goes farthest west



Demo 3: Heart of Madison

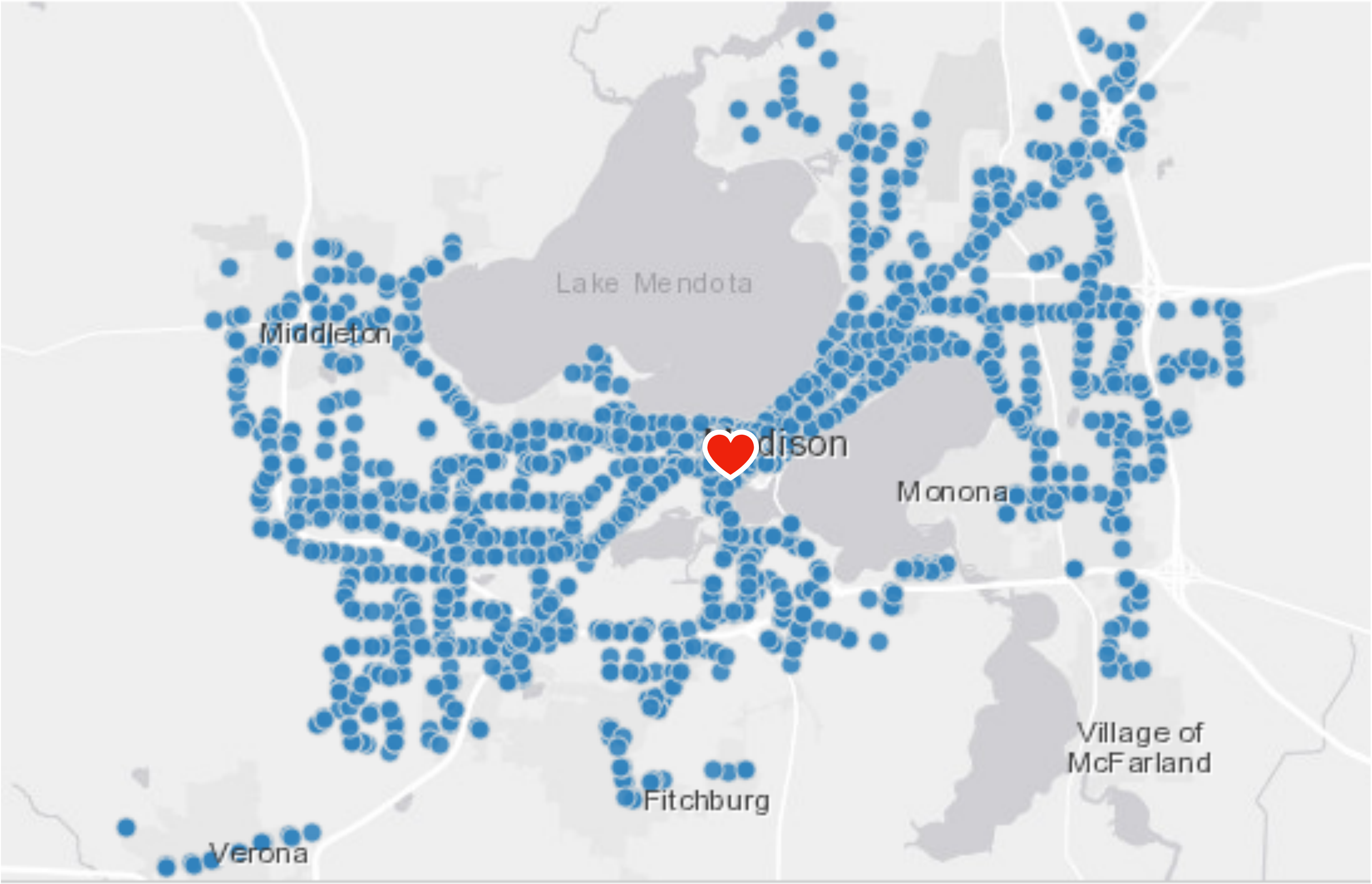
Goal: what is the central-most location of all bus pickups?

Input:

- bus.db

Output:

- a latitude and longitude



Demo 4: Fifa

Goal: load Fifa.csv to a SQLite DB, then query it

Queries:

- who are the youngest players?
- who are the oldest players?
- who are the five oldest players?
- how many players are from Brazil?
- who are the oldest players from Brazil?
- who are the 5 oldest players from Brazil?
- what percent of leagues have players from Brazil? DISTINCT

Demo 5: Vocabulary Quiz

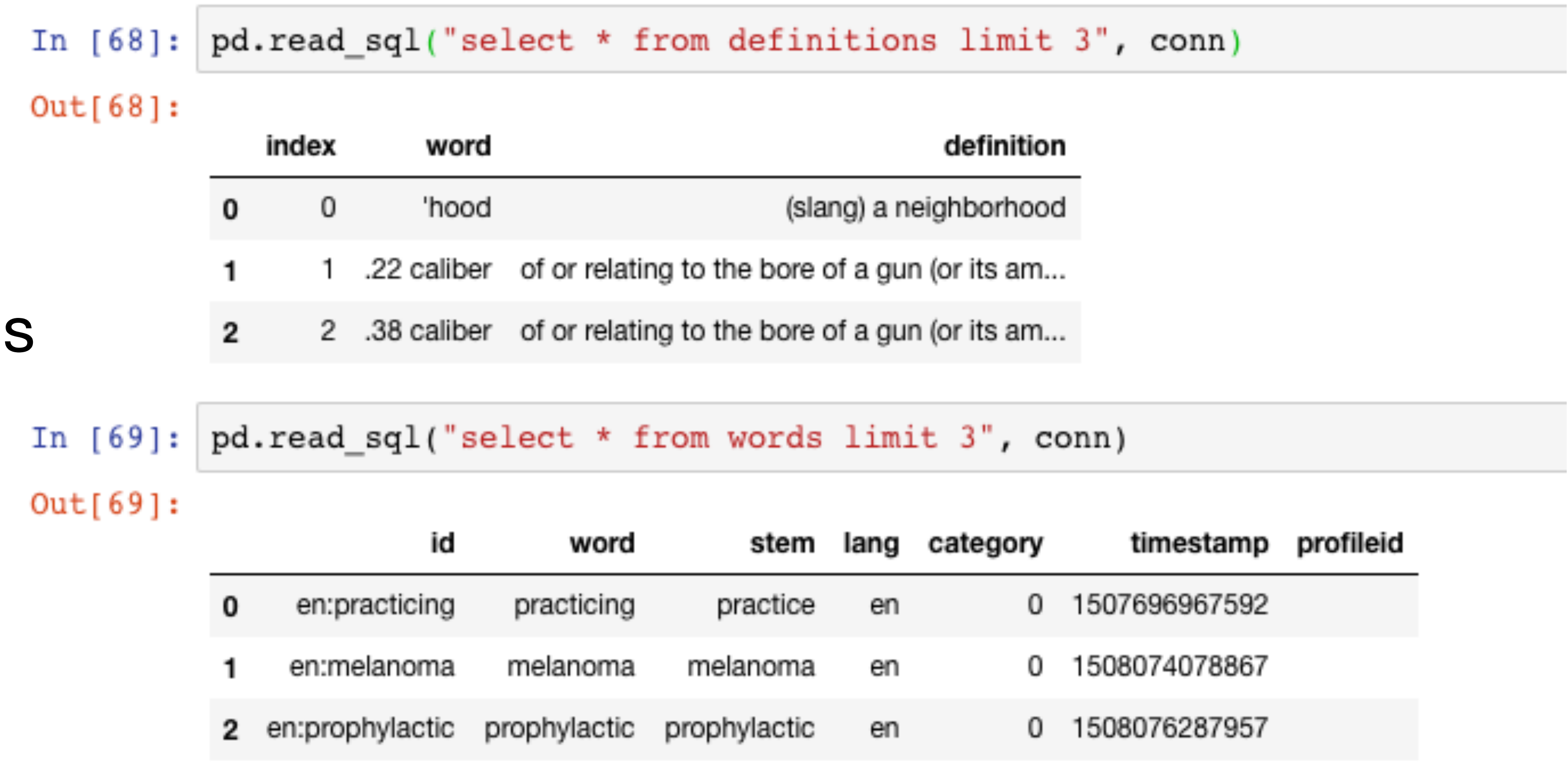
Goal: quiz user on words looked up while reading a Kindle

Input (vocab.db):

- table of kindle words lookups
- table of definitions

Output:

- random word
- real definition
- fake definitions



```
In [68]: pd.read_sql("select * from definitions limit 3", conn)
Out[68]:
```

	index	word	definition
0	0	'hood	(slang) a neighborhood
1	1	.22 caliber	of or relating to the bore of a gun (or its am...
2	2	.38 caliber	of or relating to the bore of a gun (or its am...

```
In [69]: pd.read_sql("select * from words limit 3", conn)
Out[69]:
```

	id	word	stem	lang	category	timestamp	profileid
0	en:practicing	practicing	practice	en	0	1507696967592	
1	en:melanoma	melanoma	melanoma	en	0	1508074078867	
2	en:prophylactic	prophylactic	prophylactic	en	0	1508076287957	