database and SQL

Trang Nguyen
Northwestern Libraries
Research and Learning Services

roadmap

time	topic
8:30 – 9:30am	introduction
9:30 – 10:00am	query & filter data
10:00 – 10:15am	**break**
10:15 – 11:30am	query & filter data (cont.)
11:30 – 12:00am	Q & A
12:00 – 1:00pm	**lunch**
1:00 – 2:30pm	create & modify data
2:30 – 3:15pm	**break**
3:15 – 4:00pm	SQL and R

getting to know each other

- Get together with the person next to you to create a pair. Each pair will be given a number.
- Interview each other:
  - ❑What did you do before NU?
  - ❑Why do you want to pursue this degree?
  - ❑If you can be an animal, what will you be? And why?

introduction

what is a database?
SQL and PostgreSQL
data type

what is a database?

unstructured data

First name	Last name	DOB

structured data

First name	Last name	Occ

First name	Last name	DOB

database = a **set** of structured data

scale of storage → complexity →

← accessibility

type of storage

flat file

- text file or spreadsheet
- simple data storage
- not scalable

XML

- tree structure
- more advanced data storage
- somewhat scalable

relational db

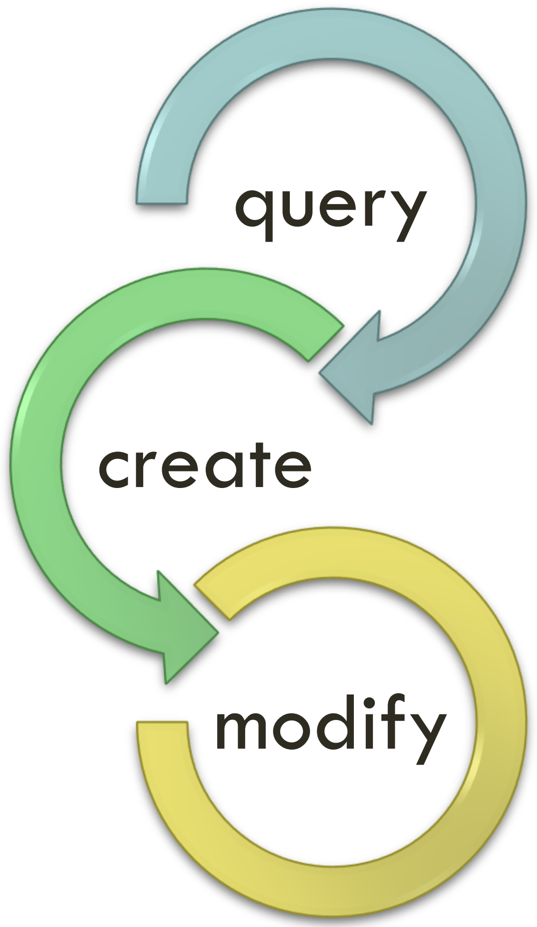
- using linked tables
- highly structured
- most advanced data storage
- highly scalable

***__short-term__* storage**
*used for **__transfer__** of data*

***__long-term__* storage**
*used for **__storage__** of data*

what to do with a db?

studentID	primary key	stored in 1 column | naturally occurred
indexingID	surrogate key	stored in 1 column | has to be created
name & DOB	composite key	stored in >1 column, same table | created by combining keys
studentID & courseID	compound key	stored in >1 column, different tables | created by combining keys



other components

trigger

- “if X happens, do Y”
- safety measure

function

- define a calculation

procedure

- a series of tasks

view

- predefined view of the data

sequence

- automatic incremental series

services

- reporting
- recovery
- others

data type

boolean

- true/false/null

character

- **char(n)**
  varchar(n): fixed-length, w/ or w/o space padded
- **text:** unlimited character

numeric

- integers: **int**
- floating-point numbers: **float(n)**, **numeric(p,s)**

temporal

- date
- time
- interval
- Others

SQL: why should you care?

robust
capability

increase
marketability

coursework
requirement

PostgreSQL

- a free and open source software to run SQL
- designed to be extensible
  - users can develop plugin to enhance functioning
- multi-version concurrency control, allowing concurrent performance

set up

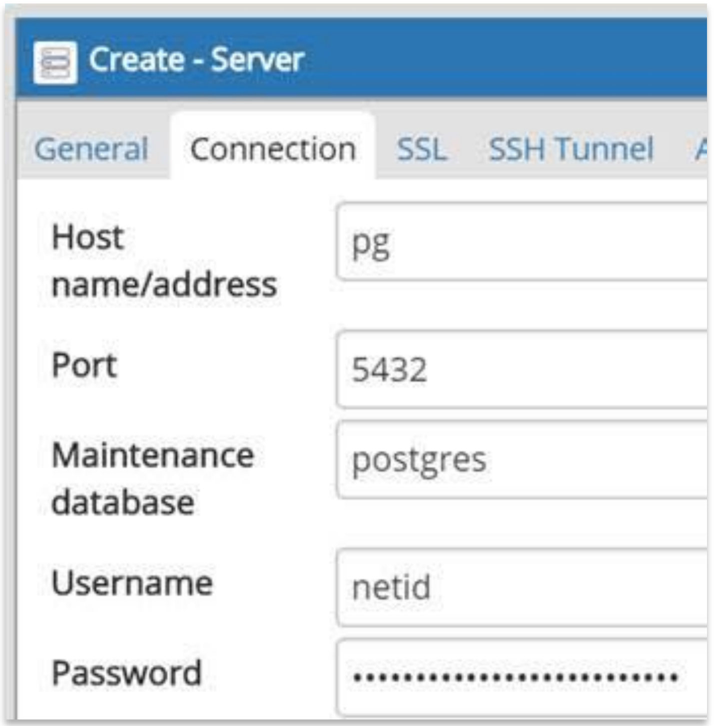
1

Connect to a remote desktop

- Open Remote Desktop Connection
- Connect to: ts2.lab.analytics.northwestern.edu
- Username: **mcc/netID**

PgAdmin

- Open PgAdmin from desktop
- Right-click Server\Create server\Pick "Connection"

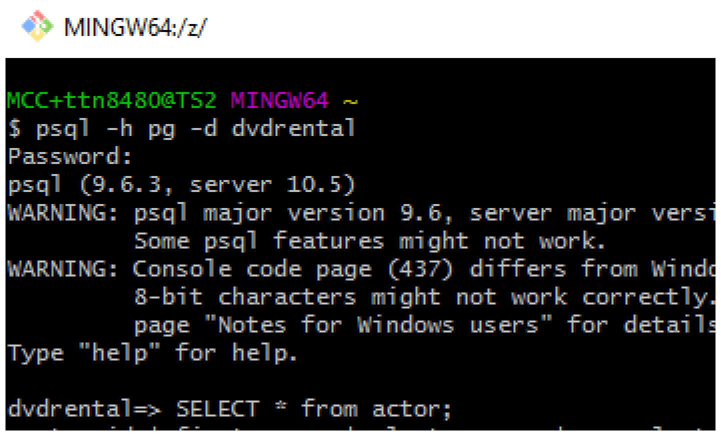


Host name/address	pg
Port	5432
Maintenance database	postgres
Username	netid
Password	••••••••••••••••••••••••••

2

Git Bash

- Open Git Bash
- Type: $ psql -h pg -d dvdrental



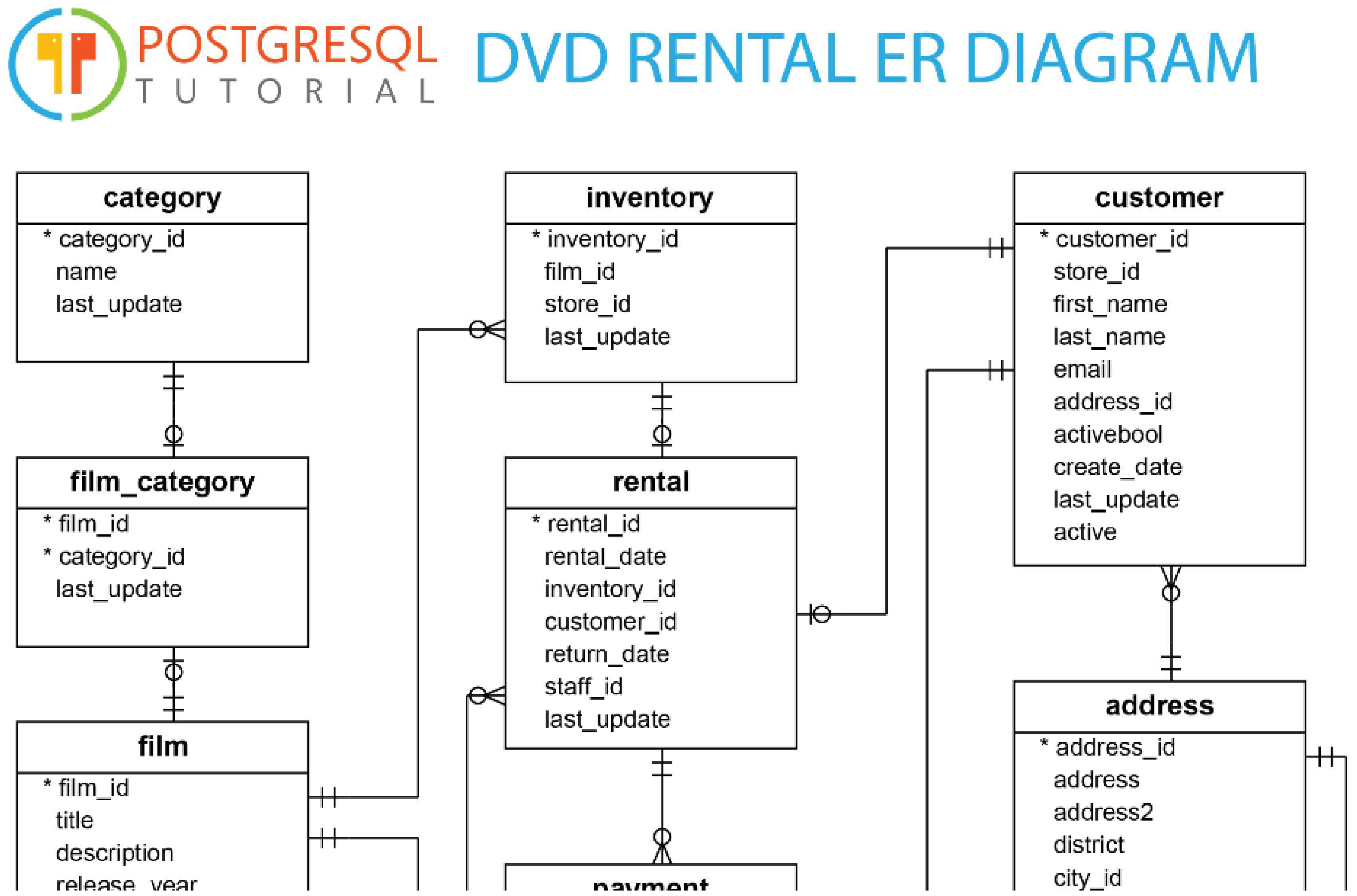
```
MCC+ttn8480@TS2 MINGW64 ~
$ psql -h pg -d dvdrental
Password:
psql (9.6.3, server 10.5)
WARNING: psql major version 9.6, server major versi
         Some psql features might not work.
WARNING: Console code page (437) differs from Windo
         8-bit characters might not work correctly.
         page "Notes for Windows users" for details
Type "help" for help.

dvdrental=> SELECT * from actor;
```

query & filter data

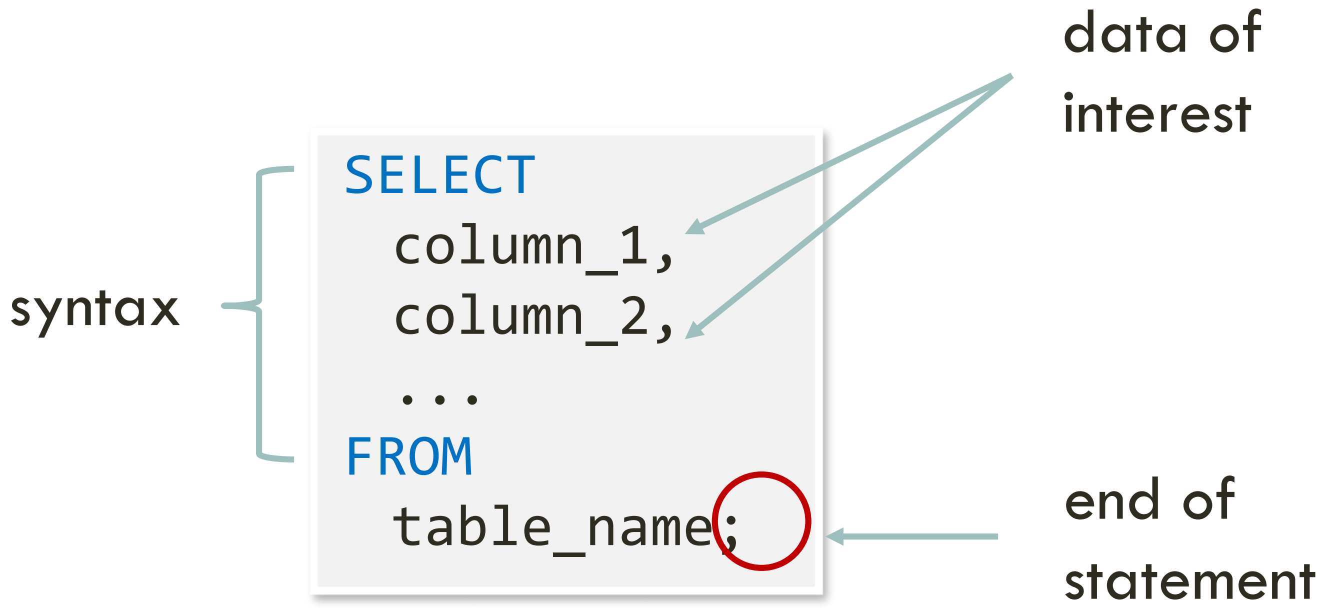
database introduction
create script
query & filter data

dvdrental db



- 15 tables
- 1 trigger
- 7 views
- 8 functions
- 1 domain
- 13 sequences

writing a statement



```
SELECT
  column_1,
  column_2,
  ...
FROM
  table_name;
```

query & filter data

ORDER BY

```
SELECT
  column_1,
  column_2,
  ...
FROM
  table_name
ORDER BY
   column_1 ASC,
   column_2 DESC;
```

WHERE

```
SELECT
  column_1,
  column_2,
  ...
FROM
  table_name
WHERE
   condition_1,
   condition_2
   ...;
```

__Logical operations:__

=, >, <, >=, <=

<> or !=

AND, OR

query data

LIMIT & OFFSET

```
SELECT
  column_1,
  column_2,
  ...
FROM
  table_name
LIMIT n
OFFSET m
```

IN

```
WHERE
        column_1 IN (value_1,value_2)
        column_2 IN
                    (
                    SELECT
                            column_x
                    FROM
                            table_name
                    );
```

query data

BETWEEN & LIKE

```
WHERE
     column_3 BETWEEN value_1
     AND value_2
WHERE
     column_4 LIKE ‘string%’
```

query data

GROUP BY

```
SELECT
  column_1,
  aggregate_function(column_2)
FROM
  table_name
GROUP BY
  column_1
```

HAVING

```
SELECT
 column_1,
 aggregate_function (column_2)
FROM
 tbl_name
GROUP BY
 column_1
HAVING
 condition;
```

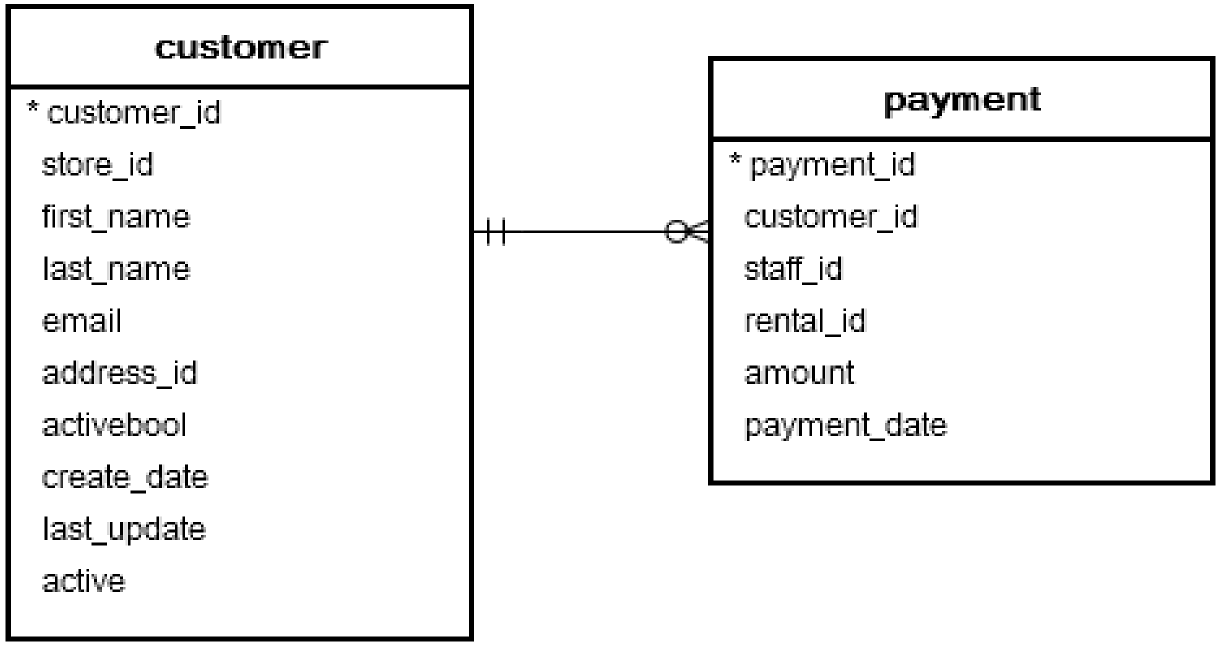
create & modify data

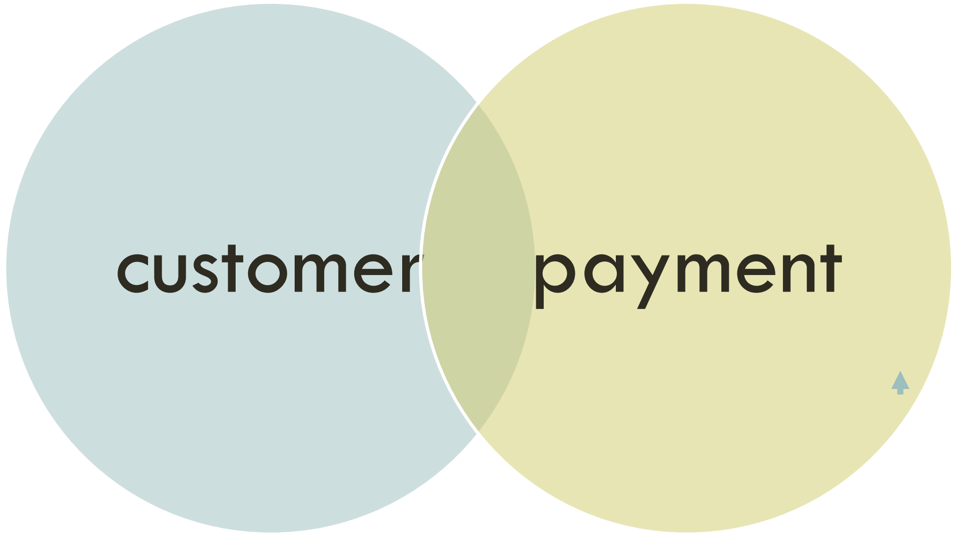
join tables

create tables

modify tables

join tables

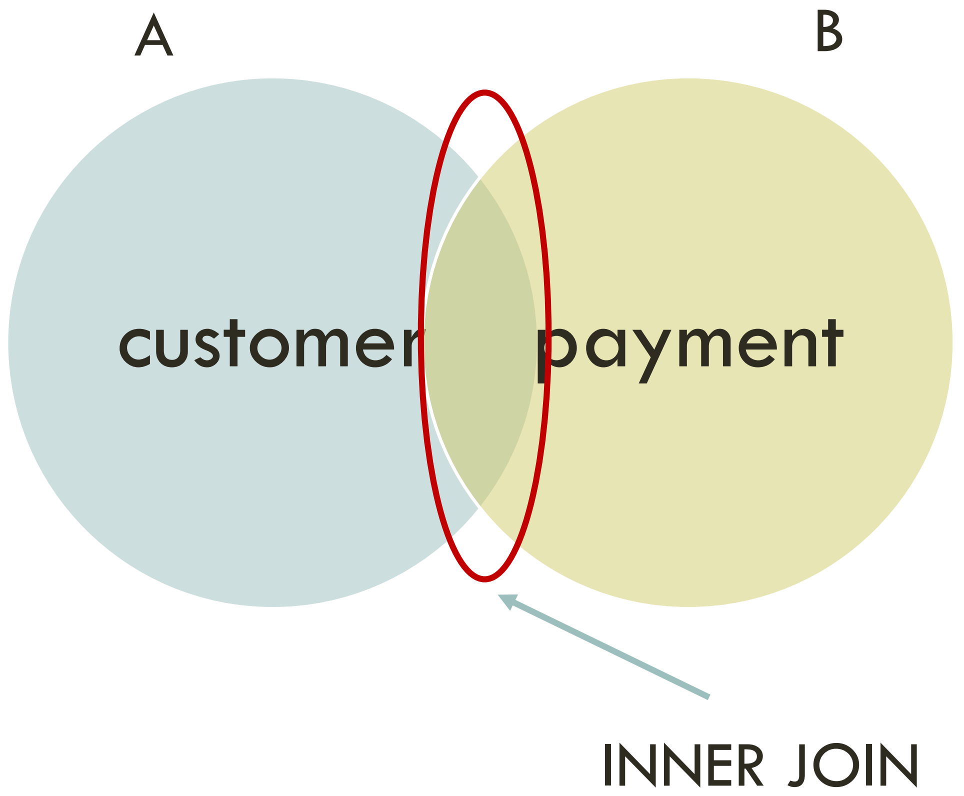

customer
* customer_id
store_id
first_name
last_name
email
address_id
activebool
create_date
last_update
active
payment
* payment_id
customer_id
staff_id
rental_id
amount
payment_date


customer
payment

join tables

INNER JOIN

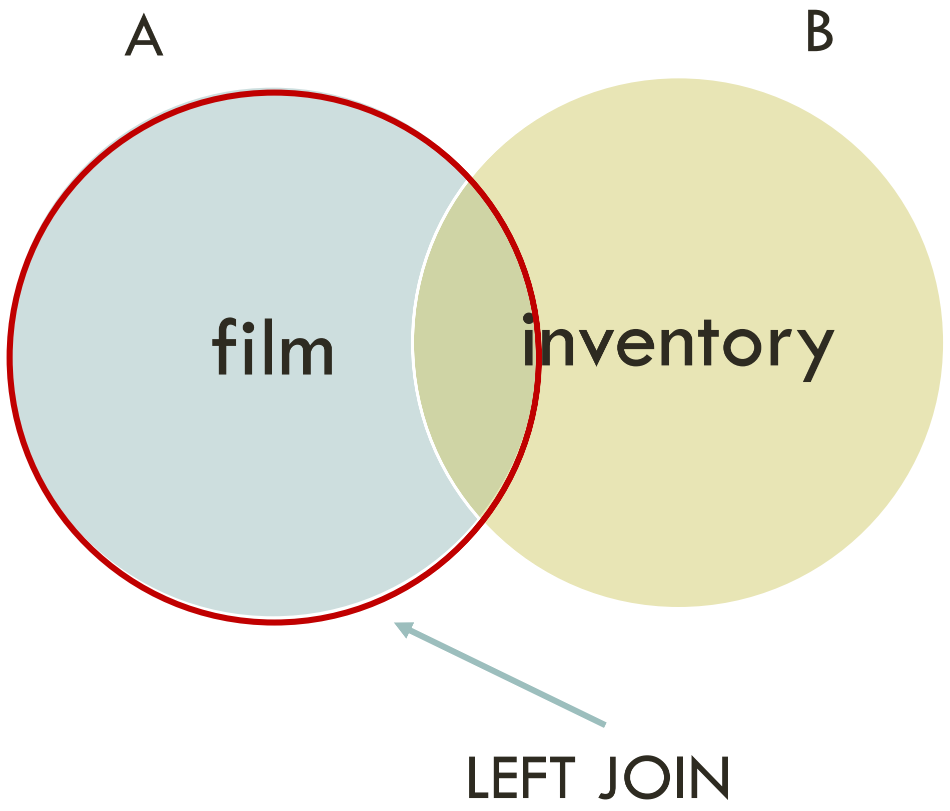
```
SELECT
 A.primarykey_a,
 A.column_1,
 B.c2
FROM
 A
INNER JOIN B ON
 A.primarykey_a = B.foreignkeya;
```



join tables

LEFT JOIN

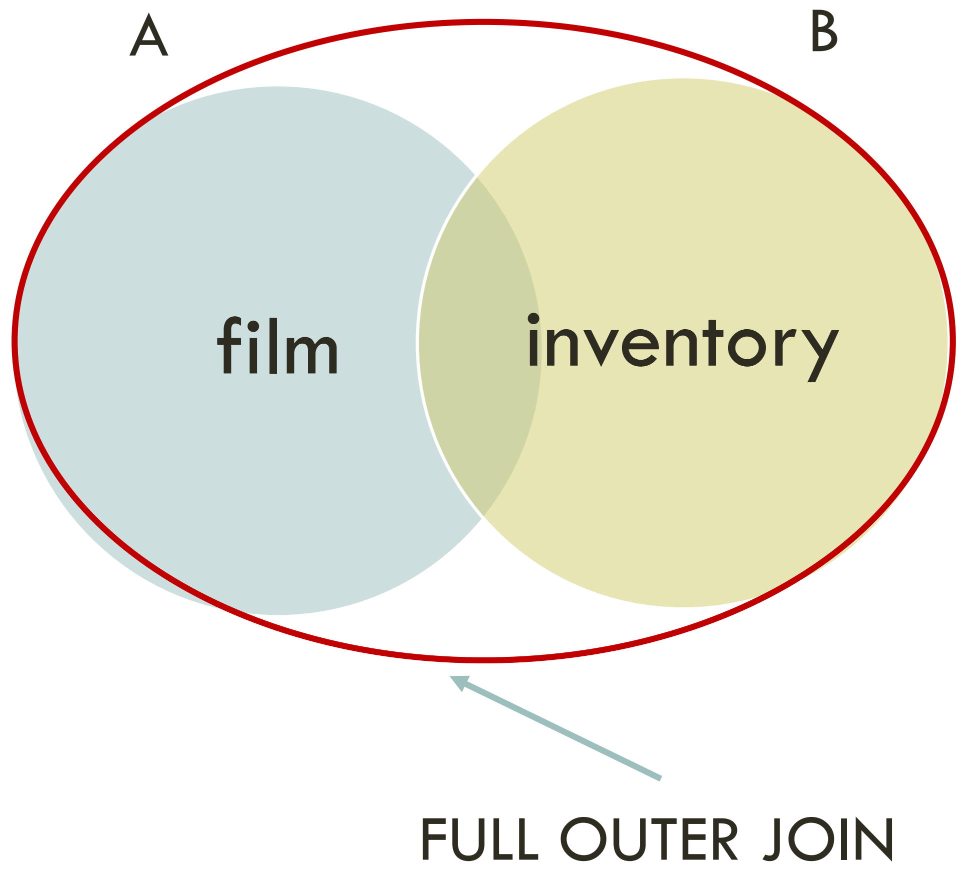
```
SELECT
 A.primarykey_a,
 A.column_1,
 B.c2
FROM
 A
LEFT JOIN B ON
 A.primarykey_a = B.foreignkeya;
```



join tables

FULL OUTER JOIN

```
SELECT
 *
FROM
 A
FULL (OUTER) JOIN B ON
 A.primarykey_a = B.foreignkeya;
```



create tables

CREATE TABLE

```
CREATE TABLE
   account (
             user_id serial PRIMARY KEY,
             username VARCHAR (50) UNIQUE NOT NULL,
             password VARCHAR (50) NOT NULL,
             email VARCHAR (355) UNIQUE NOT NULL,
             created_on TIMESTAMP NOT NULL,
             last_login TIMESTAMP
           );
```

create tables

SELECT... INTO

```
SELECT
    column_list
INTO
    new_table_name
FROM
    table_name
WHERE
    condition;
```

modify tables

ALTER TABLE

```
ALTER TABLE table_name + ...
```

```
ADD COLUMN new_column TYPE;
```

```
DROP COLUMN column_name;
```

```
RENAME COLUMN old_name TO new_name;
```

```
ADD CONSTRAINT constraint_name constraint_def;
```

modify tables

ALTER TABLE

```
ALTER TABLE table_name + ...
```

```
ALTER COLUMN column_name [SET DATA] TYPE new_data_type;
```

```
RENAME TO new_table_name;
```

other topics

import & export

SQL and R

import and export csv files

COPY statement

```
/*first create a table*/

COPY
       table_name
FROM
       ‘file_path’
       DELIMITER ',' CSV HEADER;
```

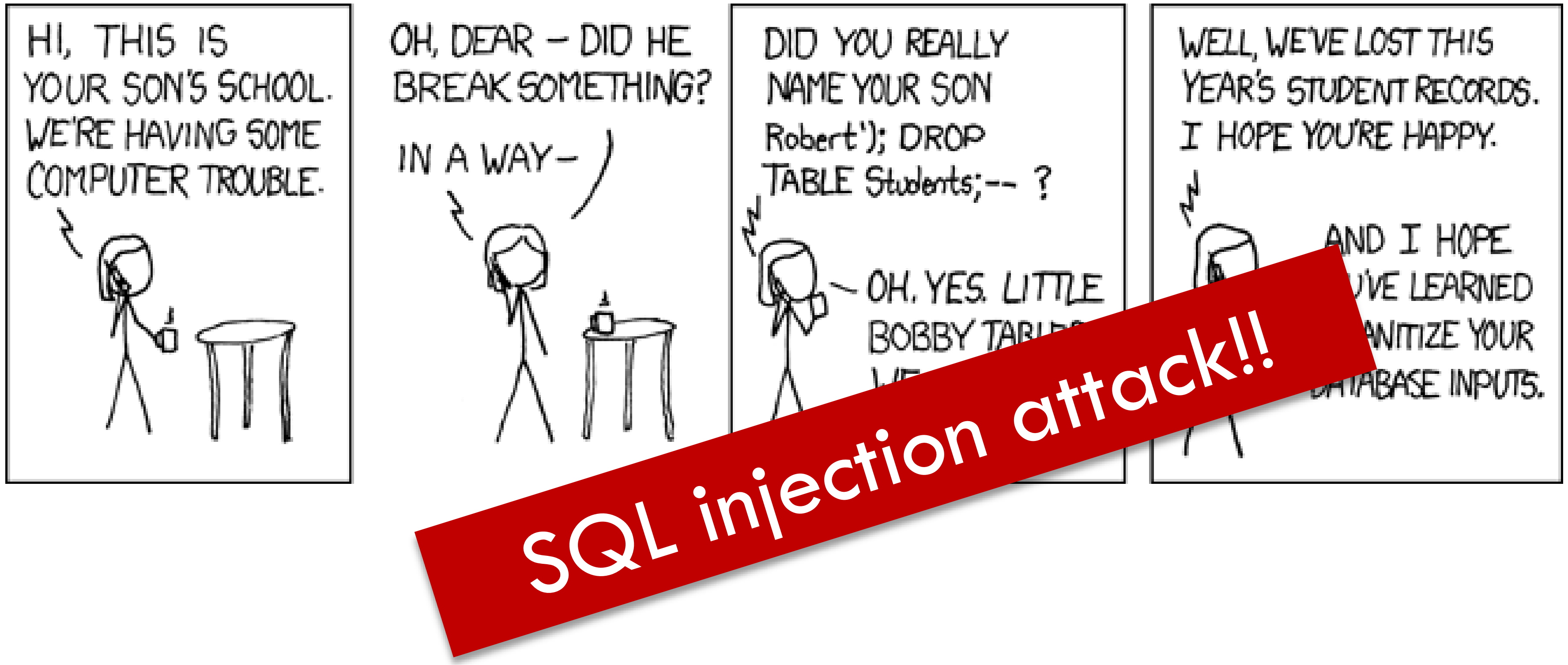
```
/*first create a table*/

COPY
       table_name
TO
       ‘file_path’
       DELIMITER ',' CSV HEADER;
```

SQL and R

```
library(RSQLite)
connection <- dbConnect(SQLite(), “dvdrental.db")
results <- dbGetQuery(connection,
                      "SELECT customer.first_name, customer.last_name
                      FROM customer;“
                      )
print(results)
dbDisconnect(connection)
```

SQL and R


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TAB
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE VE LEARNED ANITIZE YOUR ATABASE INPUTS.
SQL injection attack!!

SQL and R

```
# list table in a database
connection <- dbConnect(SQLite(), “dvdrental.db")
dbListTables(connection)

# view columns in a tables
dbListFields(connection, “film")

# read an entire table as a dataframe
dbReadTable(connection, "Person")

# write a table to a database
dbWriteTable(connection, "iris", iris, row.names = FALSE)
head(dbReadTable(connection, "iris"))
```

resources

PostgreSQL cheat sheet: http://www.postgresqltutorial.com/wp-content/uploads/2018/03/PostgreSQL-Cheat-Sheet.pdf

Psql cheat sheet: http://www.postgresonline.com/downloads/special_feature/postgresql83_psql_cheatsheet.pdf

PostgreSQL exercise: https://pgexercises.com/

Intermediate PostgreSQL: https://www.dataquest.io/blog/sql-intermediate/

get in touch

Email: datalibrarian@northwestern.edu for questions, or to set up a consultation.

Good luck!