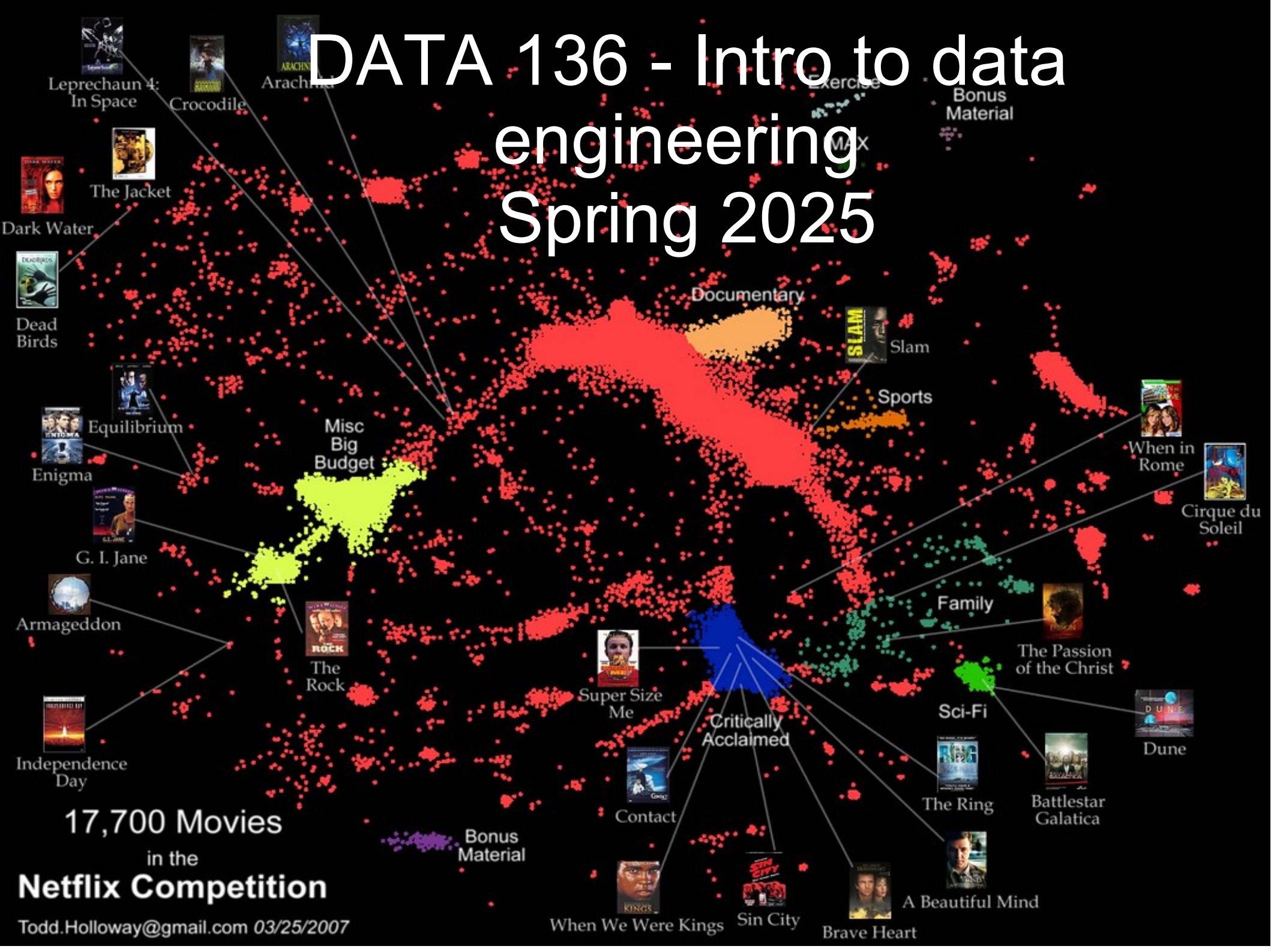
DATA 136 - Intro to data engineering Spring 2025



SSH/ SCP is cooler than you think

```
cp <source> <destination>

scp <source>  user@host:<destination>

scp user1@host1:<src>  user2@host2:<dest>
```

SSH/ SCP is cooler than you think

cp <source> <destination>

scp <source> user@host:<destination>

scp user1@host1:<src> user2@host2:<dest>

Change ~/.ssh/config to include

```
Host cs
Hostname linux.cs.uchicago.edu
User grover
Identityfile /Users/grover/.ssh/id_rsa_cs
```

Now you can publish files to the web in one line:

scp <file> cs:webdir

Q: What does a database do?

Q: What does a database do?

Put data in...
Read (subsets, relevant) data out...

Remove / update data...

A very simple "database": python dictionary

```
mydict = {}
mydict["HOME"] = "/Users/wltrimbl"
mydict["USER"] = "wltrimbl"
mydict["apikey"]
   = "72ca5c93acd491a7a757ed28483ffce8"
del mydict["HOME"]
mydict["SHELL"] = "/bin/bash"
mydict["SHELL"] = "/bin/zsh"
```

A very simple "database": python dictionary

```
mydict = {}
mydict["HOME"
mydict["USER"
mydict["apike
    = "72ca5c9
del mydict["H
mydict["SHELL
mydict["SHELL
```

Rule for adding new keys
Rule for removing keys
Rule for updating keys
Rule for handling duplicate keys

Retrieve keys by name

This is pretty lightweight; software configuration often looks kind of like this.

A very simple "database": python dictionary

```
mydict = {}
mydict["HOME"
mydict["USER"
mydict["apike
    = "72ca5c9
del mydict["H
mydict["SHELL
mydict["SHELL
```

No validation of input

Datatypes could be anything

called a key-value store

Version control databases

```
% git log
commit 803a48b76394 (HEAD -> hw_0)
Author: Grover <grover@WA.local>
Date:   Mon Sep 30 17:50:47 2024 -0500
    Added names.txt

commit 9f7bea2a6864 (origin/hw_0)
Author: Grover <grover@WA.local>
Date:   Mon Sep 30 14:12:54 2024 -0500
    added names.csv

commit 7e834d898c16 (main)
Author: Grover <grover@WA.local>
Date:   Mon Sep 30 11:58:44 2024 -0500
    nobel-prize-laureates.csv

commit 2882dc45a1746
Author: Grover <grover@WA.local>
Date:   Mon Sep 30 10:49:01 2024 -0500
    removed unneeded_data

commit c7397952aa35(origin/main, origin/HEAD)
Author: github-classroom[bot] <66690702+github-classroom[bot]@users.noreply.
github.com>
Date:   Mon Sep 30 02:41:01 2024 +0000
    Initial commit
% git push origin hw_0
```

Version control databases record changes.

The fields they record include human name, email, date, **the content of the changes**, and a **commit message**.

This is a hassle. Not only do I have to do the work, but I have to write a message saying that I did the work? Kill me now.

Q: Why bother?

Shell: two-letter commands

- cp <source> <destination>
- cp <source1> <source2> ... <destination>
  - if desination is a directory
- mv <source> <destination>
  - move or rename; does not check whether destination already exists (clobbers)
- cd <directory name>
- rm <file1> [<file2>] [<file3] ...
  - silenty destroy
- ls [<directory name> | <list of one or more files>]

Shell: plumbing

- Shell commands produce output in a stream called "standard out" and those that accept input can accept "standard in"
- ls | grep a # takes output of ls (one line per filename) and feeds it to grep, which prints only lines containing the pattern "a"
- Many commands will take a filename as an argument, or, if no filename is provided, standard in, but many behave slightly differently when the filename is available (for instance, wc and grep show filenames in output)

Relational Databases

50 years ago, computing decided to separate "programming" business logic from data handling, and built dedicated software to store, retrieve, manipulate, and control access to "data".

Asked engineer: why? answer was "when the hassle of running a database solves more of your problems than it creates"

Dozens of "implementations" : software stacks that store data, usually understand SQL DDL and DML.

These lie behind everything in the modern world.

Relational Databases take their name from relational algebra

- Tables are called relations
- Rows are called tuples
- Columns are called attributes
- Operations like select, project, logical operators like and and or, set operations like cross product and intersection.

- Implementations of database management systems resting on this mathy framework dominate computing.

What does it do?

Standardizes interaction between programs and databases.

SQL is unlike other languages: designed to be written by programs!!!

SQL: Originally “SEQUEL” from IBM’s **System R** prototype.

→ Structured English Query Language.

→ Adopted by Oracle in the **1970s**.

IBM releases DB2 in **1983**.

ANSI Standard in **1986**. ISO in **1987**

→ Structured Query Language

Invented for electronic banking, order processing, supply chain, etc.

SQL has two essential parts (sub-languages?):

Data Definition Language (DDL)

Data Manipulation Language (DML)

SQL has two essential parts:

Data Definition Language (DDL)

specifies how the data are arranged, including

- the schema for each table
- the domain of values allowed for each attribute.

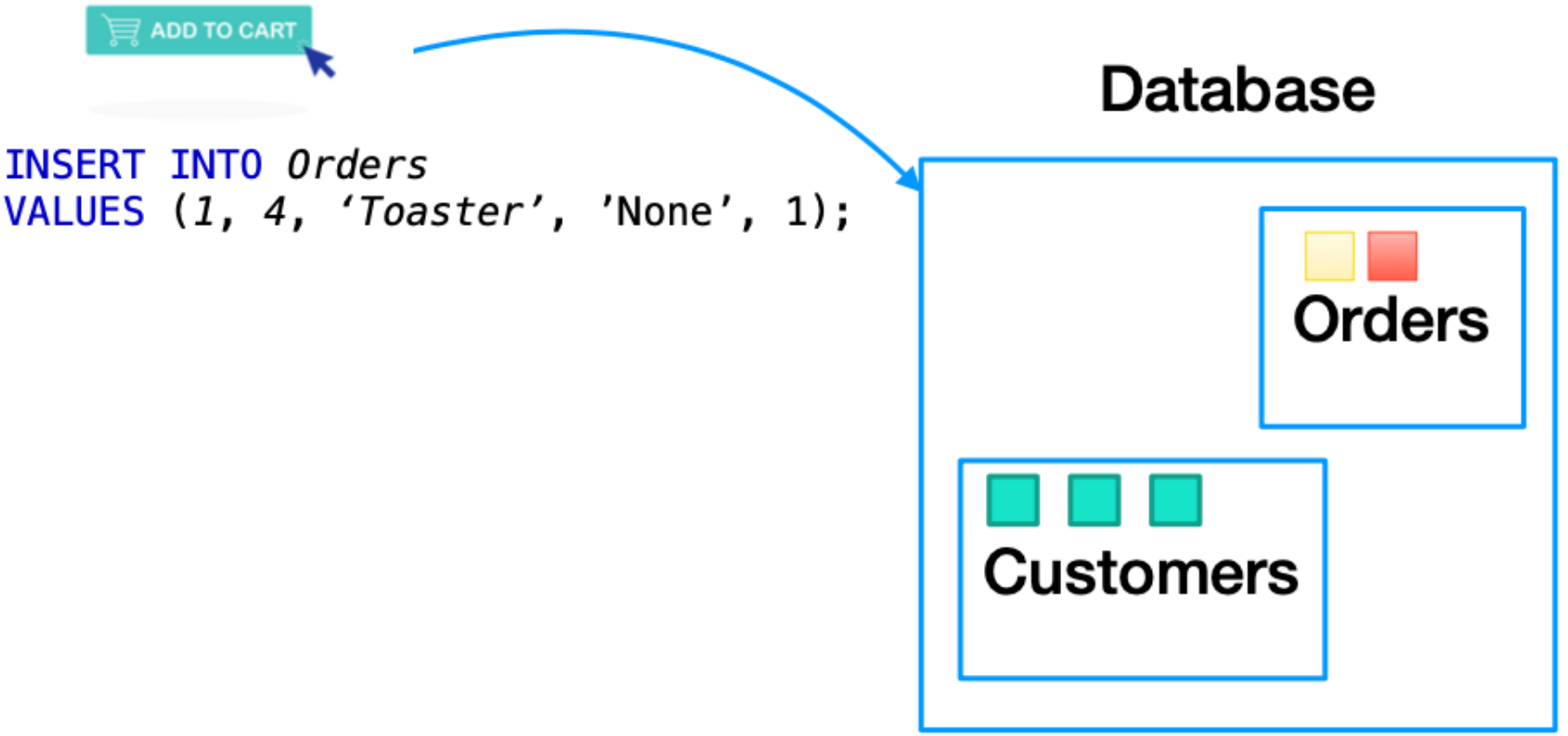
Database schema...?

- A database schema defines how data is organized within a relational database; this [includes] logical constraints such as, table names, fields, data types and the relationships between these entities -- IBM.
- The database schema is the structure of a database described in a formal language supported typically by a relational database management system (RDBMS). --Wikipedia

DDL examples

```
CREATE TABLE Customers ( CustomerID int,
        LastName varchar(255),
        FirstName varchar(255),
        Address varchar(255),
        City varchar(255)
);
```

```
CREATE TABLE Orders (
      OrderID int,
      CustomerID int,
      Product varchar(255),
      ShippingInst varchar(255),
      Qty int
);
```


ADD TO CART
INSERT INTO Orders
VALUES (1, 4, 'Toaster', 'None', 1);
Database
Orders
Customers

DML examples

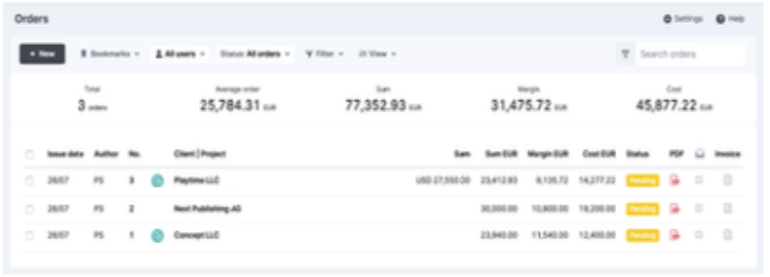
```
INSERT INTO Customers VALUES (1, 'John', 'Smith',
'302 N Michigan', 'Chicago');

INSERT INTO Customers VALUES (2, 'Jenny', 'Qiang',
'1802 S Prairie', 'Chicago');
```

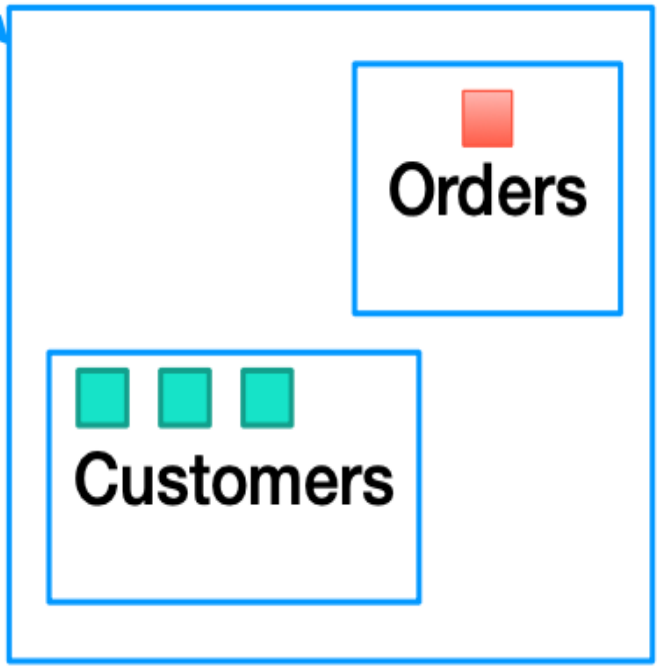
```
INSERT INTO Orders VALUES (1, 2, 'Bug deflector',
'', 1);

INSERT INTO Orders VALUES (2, 2, 'Insect repellent',
'', 2);

INSERT INTO Orders VALUES (3, 1, 'Binoculars', '',
1);
```




```
Select *
FROM Orders, Customers
WHERE Orders.CustomerId = 5
```



SQL has two essential parts:

DDL

CREATE
ALTER
DROP

DML

SELECT
INSERT
UPDATE
DELETE

SQL data types...

char(n). Fixed length character string, with user-specified length *n.*
varchar(n). Variable length character strings, with user-specified

maximum length *n.*

int. Integer (a finite subset of the integers that is machine-dependent).

smallint. Small integer (a machine-dependent subset of the integer domain type).

numeric(p,d). Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.

real, double precision. Floating point and double-precision floating point numbers, with machine-dependent precision.

float(n). Floating point number, with user-specified precision of at least *n* digits.

`cat <filename>` # streams contents of filename to STDOUT

`wc <filename>` # count lines, words, characters in filename

`cut -f N[,N2][,N3-N4] [-d SEP]` # split file on SEP and output (to STDOUT) only specified columns

`grep <pattern> [<filename>]`

Some pretty good resources

- Shell: https://swcarpentry.github.io/shell-novice/
- Git: https://swcarpentry.github.io/git-novice/
- Python: https://developers.google.com/edu/python
- SQL https://sql.js.org/examples/GUI/
- SQL https://sqliteonline.com/