Database Class 1

Goal for today: You'll be comfortable opening and using database management software. You'll select particular records from larger data to find patterns for stories.

A database is a collection of tables and views of tables and collections of instructions
We'll start with one table, but soon we'll be linking tables together

Software

There are many choices of software to manage databases, such as Microsoft Access, Oracle, IBM, mysql, postgresql, Microsoft sqlserver and sqlite. We'll use mysql, an open-source product with a free version.

You can use mysql in Mac, Windows or Linux.
It uses a server-client situation. The data is handled by the server portion of mysql. You manipulate it via a client. The advantage of this is flexibility. The data can be available via web browser or phone app. The server and client can also be on the same machine while you are deciding what to serve out publicly. You can download and install mysql and run it for free on your own laptop.

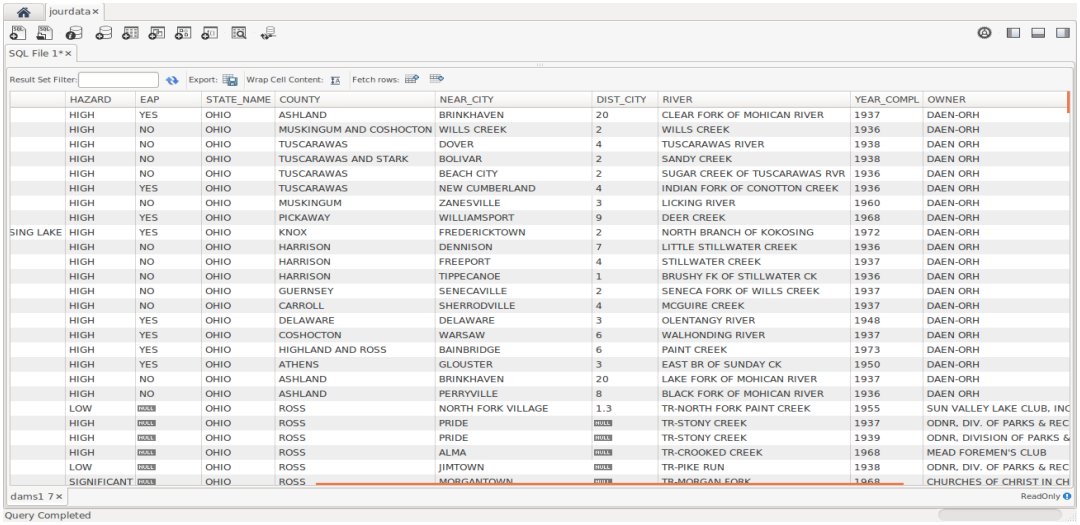
For this class, we'll use the friendly client tool on your machine. The data sits on university server. So we'll connect from your machine to the server and manipulate/access the data there.

You can download and install this same client on your laptop and access the server from anywhere.

Tables

Table Structure

- records and fields (fields also called variables), appears as rows and columns
- conceptual: each record (row) is one record
- be conscious of the unit of measurement, granularity
- field types: number, text, date, time
- this is STRUCTURED DATA



	HAZARD	EAP	STATE_NAME	COUNTY	NEAR_CITY	DIST_CITY	RIVER	YEAR_COMPL	OWNER
	HIGH	YES	OHIO	ASHLAND	BRINKHAVEN	20	CLEAR FORK OF MOHICAN RIVER	1937	DAEN-ORH
	HIGH	NO	OHIO	MUSKINGUM AND COSHOCTON	WILLS CREEK	2	WILLS CREEK	1936	DAEN-ORH
	HIGH	NO	OHIO	TUSCARAWAS	DOVER	4	TUSCARAWAS RIVER	1938	DAEN-ORH
	HIGH	NO	OHIO	TUSCARAWAS AND STARK	BOLIVAR	2	SANDY CREEK	1938	DAEN ORH
	HIGH	NO	OHIO	TUSCARAWAS	BEACH CITY	2	SUGAR CREEK OF TUSCARAWAS RVR	1936	DAEN ORH
	HIGH	YES	OHIO	TUSCARAWAS	NEW CUMBERLAND	4	INDIAN FORK OF CONOTTON CREEK	1936	DAEN ORH
	HIGH	NO	OHIO	MUSKINGUM	ZANESVILLE	3	LICKING RIVER	1960	DAEN-ORH
	HIGH	YES	OHIO	PICKAWAY	WILLIAMSPORT	9	DEER CREEK	1968	DAEN-ORH
SING LAKE	HIGH	YES	OHIO	KNOX	FREDERICKTOWN	2	NORTH BRANCH OF KOKOSING	1972	DAEN-ORH
	HIGH	NO	OHIO	HARRISON	DENNISON	7	LITTLE STILLWATER CREEK	1936	DAEN ORH
	HIGH	NO	OHIO	HARRISON	FREEPORT	4	STILLWATER CREEK	1937	DAEN-ORH
	HIGH	NO	OHIO	HARRISON	TIPPECANOE	1	BRUSHY FK OF STILLWATER CK	1936	DAEN ORH
	HIGH	NO	OHIO	GUERNSEY	SENECAVILLE	2	SENECA FORK OF WILLS CREEK	1937	DAEN ORH
	HIGH	NO	OHIO	CARROLL	SHERRODVILLE	4	MCGUIRE CREEK	1937	DAEN-ORH
	HIGH	YES	OHIO	DELAWARE	DELAWARE	3	OLENTANGY RIVER	1948	DAEN-ORH
	HIGH	YES	OHIO	COSHOCTON	WARSAW	6	WALHONDING RIVER	1937	DAEN ORH
	HIGH	YES	OHIO	HIGHLAND AND ROSS	BAINBRIDGE	6	PAINT CREEK	1973	DAEN-ORH
	HIGH	YES	OHIO	ATHENS	GLOUSTER	3	EAST BR OF SUNDAY CK	1950	DAEN-ORH
	HIGH	NO	OHIO	ASHLAND	BRINKHAVEN	20	LAKE FORK OF MOHICAN RIVER	1937	DAEN-ORH
	HIGH	NO	OHIO	ASHLAND	PERRYVILLE	8	BLACK FORK OF MOHICAN RIVER	1936	DAEN ORH
	LOW	NULL	OHIO	ROSS	NORTH FORK VILLAGE	1.3	TR-NORTH FORK PAINT CREEK	1955	SUN VALLEY LAKE CLUB, INC
	HIGH	NULL	OHIO	ROSS	PRIDE	NULL	TR-STONY CREEK	1937	ODNR, DIV. OF PARKS & REC
	HIGH	NULL	OHIO	ROSS	PRIDE	NULL	TR-STONY CREEK	1939	ODNR, DIVISION OF PARKS &
	HIGH	NULL	OHIO	ROSS	ALMA	NULL	TR-CROOKED CREEK	1968	MEAD FOREMEN'S CLUB
	LOW	NULL	OHIO	ROSS	JIMTOWN	NULL	TR-PIKE RUN	1938	ODNR, DIV. OF PARKS & REC
	SIGNIFICANT	NULL	OHIO	ROSS	MORGANTOWN	NULL	TR-MORGAN FORK	1968	CHURCHES OF CHRIST IN CH

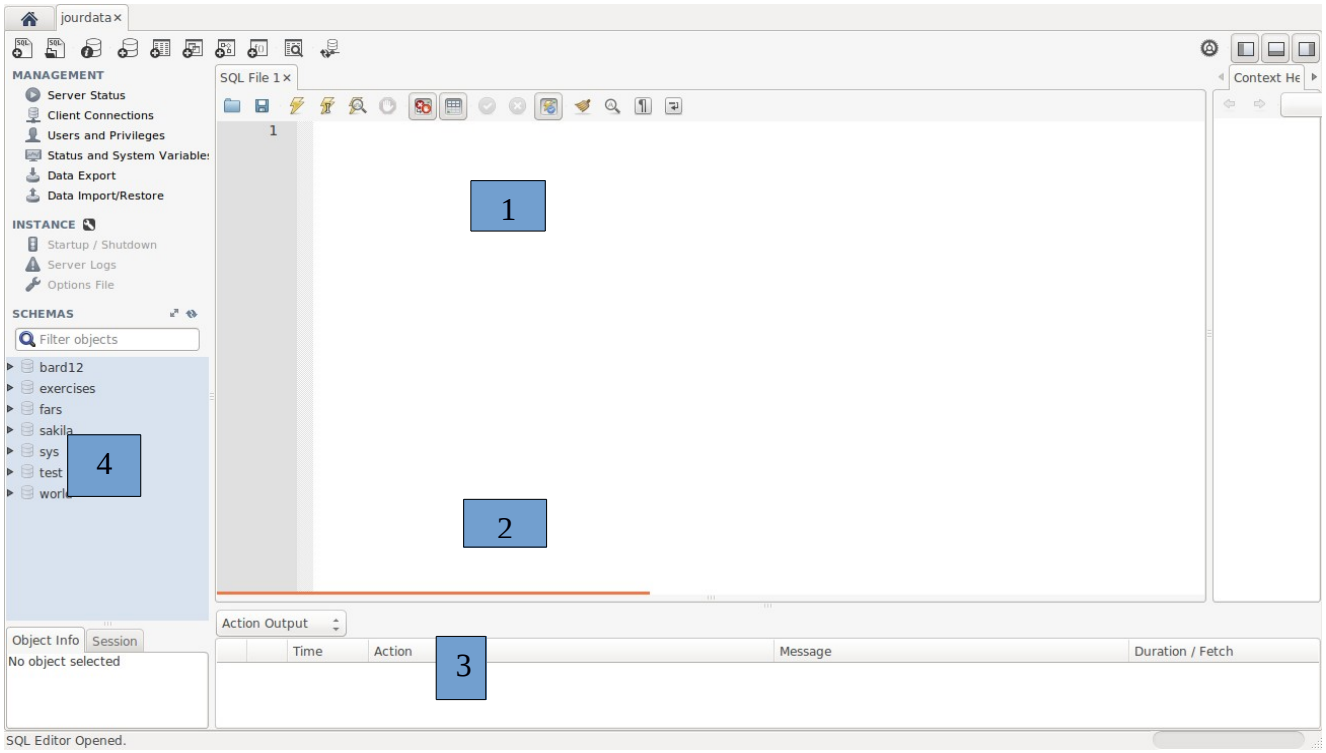
Language
There is a specific language for structured data called Structured Query Language, SQL. We'll use SQL

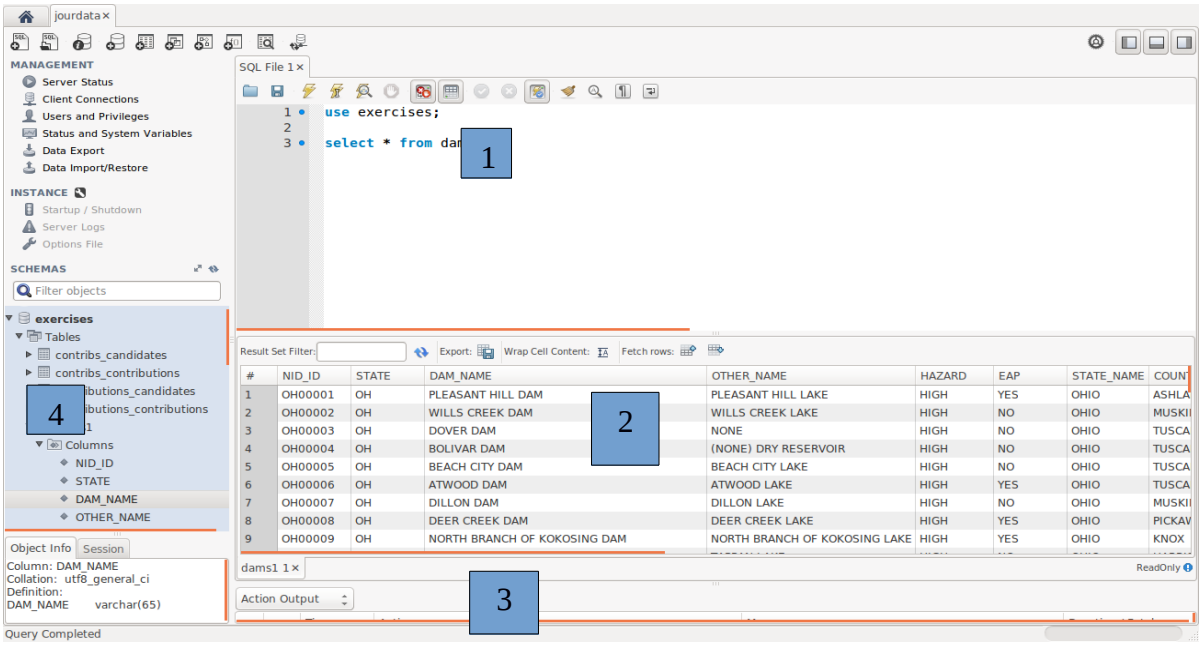
MYSQL Client
Open software mysql workbench in Mac or Windows
Create a connection to the college server.
In the connections manager – the "+" sign to add new connection
Fill out boxes: give it whatever name you want – jourdata is a suggestion

then "test connection" – should say OK. If not. Let me know.
Click OK to save it. You won't have to recreate it every time.
Click on that saved connection to start using the software.

Key parts of the software. The size of each module on the screen can be adjusted.
There are four key areas that you need to use regularly.

1 – SQL tab text area to write commands
2 – Results output area (appears after you run a query)
3 – Results status update
4 – Information browser





Queries
Pulling selected records from the data table is the first step in our data reporting.

In the SQL Script window, type first command:
Use exercises;
Execute it with the Lightning icon
Now the system knows that you will use the tables in the exercise “schema”

The first query:
Select dam_name from dams1;
Execute it with the Lighting icon.
The results have one record for each record in the data.

Add sorting with “order by.” Default is ascending “asc,” but can be descending “desc”:

```
Select dam_name
from dams1
order by dam_name;
```

Repeat with order by dam_name desc;

Getting more fields:

```
Select dam_name, state, state_name
from dams1
order by state;
```

```
Select dam_name, state, state_name
from dams1
order by state_name, dam_name;
```

To get all fields, use *

```
select *
from dams1
order by year_compl desc;
```

So the design is basic:

```
Select fieldname1, fieldname2, fieldname3
from table1
order by fieldname1, fieldname2, fieldname3
```

Filtering
Next is filtering with the “where” command to make the software only display records that meet filtering instructions. The “where” comes immediately after the “from.”

Match text:

```
Select dam_name, near_city
from dams1
where state = 'OH'
order by near_city;
```

Match a number:

```
Select dam_name, near_city, dist_city, owner
from dams1
where dist_city = 10
order by near_city;
```

Operators for matching: equals = , less than < , greater than >, less than or equal to <=, greater than or equal to >= , between xx and xx, “in” a list

```
Select dam_name, near_city, dist_city, owner
from dams1
where dist_city < 10
order by near_city;

select *
from dams1
where year_compl < 1950;
```

Matching a full date, put year-month-day:

```
Select dam_name,  insp_date
from dams1
where insp_date < '1993-1-1';
```

Using between for a range of dates:

```
Select dam_name, insp_date
from dams1
where insp_date between  '1983-01-01' and '1992-06-30'
order by insp-date desc;
```

The “in” operator with a list of values separated by a comma.

```
Select dam_name, insp_date, state, near_city
from dams1
where near_city in ('Springfield','Morgantown');
```

Be careful to have the comma outside the parenthesis.

Combining filters with “and” and “or.” You can use multiple filters are once.

```
Select dam_name, near_city, year_compl
from dams1
where year_compl < 1960 and near_city < 8;

Select dam_name, near_city, year_compl
from dams1
where near_city = 'Morgantown' or near_city = 'Springfield';

Select dam_name, near_city, year_compl
from dams1
where near_city = 'Morgantown' and near_city = 'Springfield';
```

Wildcards let you match text more flexibly. The character % can represent any characters. The “_”

matches any single character. For matching, use “like” instead of = (equals).

```
Select dam_name, near_city, dist_city, year_compl
from dams1
where near_city like 'River%';

Select dam_name, near_city, dist_city, year_compl
from dams1
where near_city like '%haven'
order by near_city desc;

select *
from dams1
where near_city like 'new%town';
```

Modifying any filter with “not” to get the opposite. For text comparison “not” is != or <>.

```
Select dam_name
from dams1
where dam_name  like 'big%';

Select dam_name
from dams1
where dam_name not like 'big%';

select dam_name, near_city, hazard
from dams1
where hazard <> 'low'
order by hazard;

select *
from dams1
where near_city not in ('Springfield','Perryville','Glouster');
```

Find missing information, search for null:

```
Select insp_date, dam_name, near_city
from dams1
where insp_date is null;
```

and the inverse

```
Select insp_date, dam_name, near_city
from dams1
where insp_date is not null;
```

Using parenthesis with multiple terms, especially when mixing “or” and “and.”

Build this example query bit-by-bit.

```
select insp_date, dam_name, near_city
from dams1
where (hazard = 'low' or hazard = 'high') and (eap = 'yes' or eap is null);
```

What kind of **filters** do we have:
Matching text, number, date
Broader matches with greater than, less than, between
Using in with a list separated by commas
Combining filters with and-or
Even broader filters with wildcards
Find missing information with “is null”
Using not-equals

Pay attention to:
data documentation
record counts
Save queries