```
In [1]: %load_ext sql
        %sql postgresql://dssg_student:password@seds-sql.csya4zsfb6y4.us-
        east-1.rds.amazonaws.com/dssg2016

Out[1]: 'Connected: dssg_student@dssg2016'
```

Introduction to Databases with SQL

Valentina Staneva

eScience Institute

vms16@uw.edu

11/30/2016

Plan for Today:

- quickly overview functionality of databases and SQL
- learn fundamental SQL commands by exploring a Seattle Crime Dataset

Data source: http://www.seattle.gov/seattle-police-department/crime-data/spd-data-sets (http://www.seattle.gov/seattle-police-department/crime-data/spd-data-sets)

Lesson source: https://github.com/valentina-s/SQL_tutorial/ (https://github.com/valentina-s/SQL_tutorial/)

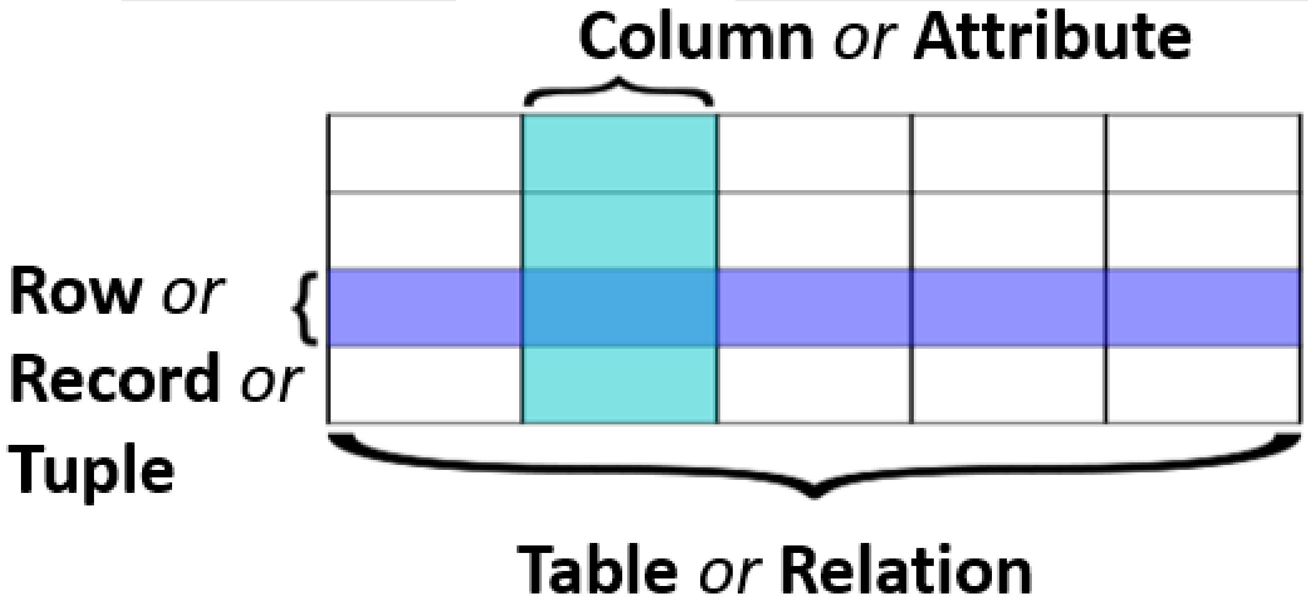
Why Databases???

- data you need is in a public database
- you want to make your data easily accessible
- you can manage the security permissions of the data
- computational load is on the server side
- you cannot load data into RAM
- certain operations are optimized in a database
- you can implement efficient scalable algorithms in a database

What is a Database?

"A database system is basically a computerized recordkeeping system -- that is, a system whose overall purpose is to maintain information and to make that information available on demand." (Date 1986)

- conventional databases use the *relational* data model



Relational Data Model:

- data are structured into row/column format

crimesID	Offense type	Offense code	Date	Location
1	tresspass	5700	2015-01-28 09:30:00	12XX Block of E Pike St
2	larceny-theft	2300	2015-02-21 08:24:21	15XX Block of Aurora St

- each record has a unique identifier (primary key)

What is SQL?

Structured Query Language

- Domain-Specific Programming Language
- not Turing Complete (cannot build a robot which can follow any instruction :()

But great for data manipulation!!!

And human-readable!!!

Creating a Table:

```
CREATE TABLE seattlecrimesincidents
    ("crimesID" int,
     "Offense type" character,
     "Offense code" int,
     "Date" timestamp,
     "Location" character);
```

crimesID	Offense type	Offense code	Date	Location

- populating the database records:

```
INSERT INTO seattlecrimeincidents VALUES

  (1,'trespass', 5700,'2015-01-28 09:30:00','12XX Block of E Pik
e St'),

  (2,'larceny-theft',2300, '2015-02-21 08:24:21','15XX Block of
Aurora St');
```

crimesID	Offense type	Offense code	Date	Location
1	tresspass	5700	2015-01-28 09:30:00	12XX Block of E Pike St
2	larceny-theft	2300	2015-02-21 08:24:21	15XX Block of Aurora St

Data in each column must be of the same type

Some common data types (https://www.postgresql.org/docs/9.4/static/datatype.html):

Name	Aliases	Description
`boolean`	`bool`	logical Boolean (true/false)
`character [(n)]`	`char [(n)]`	fixed-length character string
`date`		calendar date (year, month, day)
`double precision`	`float8`	double precision floating-point number (8 bytes)
`integer`	`int`, `int4`	signed four-byte integer
`json`		JSON data
`money`		currency amount
`timestamp [(p)] [ without time zone ]`		date and time (no time zone)
`xml`		XML data

NULL values

- missing data are a common feature of many datasets
- here the code for "tresspass" is not known so the data entry is "X"

crimesID	Offense type	Offense code	Date	Location
1	tresspass	X	2015-01-28 09:30:00	12XX Block of E Pike St
2	burglary	5710	2015-01-28 09:30:00	12XX Block of E Pike St
3	larceny-theft	2300	2015-02-21 08:24:21	15XX Block of Aurora St

NULL values

- conventionally, some value is used to represent missing data (e.g. "X" or -9999)
- relational databases introduced NULL values:
  - NULL is a state representing a lack of a value
  - NULL is not the same as zero!
  - NULL values are ignored in SELECT statements

Normalization (https://en.wikipedia.org/wiki/Database_normalization)

- minimize redundancy

Example: multiple offenses at the same time

crimesID	Offense type	Offense code	Date	Location
1	tresspass and burglary	5700 and 5710	2015-01-28 09:30:00	12XX Block of E Pike St
2	larceny-theft	2300	2015-02-21 08:24:21	15XX Block of Aurora St

INCORRECT: database will have problems searching these columns

Solution: create another row

crimesID	Offense type	Offense code	Date	Location
1	tresspass	5700	2015-01-28 09:30:00	12XX Block of E Pike St
2	burglary	5710	2015-01-28 09:30:00	12XX Block of E Pike St
3	larceny-theft	2300	2015-02-21 08:24:21	15XX Block of Aurora St

Selecting Rows:

```
SELECT *
   FROM seattlecrimeincidents
   WHERE "Offense code" = 5700;
```

- use a "WHERE" clause to select specific rows

crimesID	Offense type	Offense code	Date	Location
1	tresspass	5700	2015-01-28 09:30:00	12XX Block of E Pike St

Selecting Columns:

```
SELECT "Offense type", "Date"
   FROM seattlecrimeincidents;
```

- use a comma separated list to select specific columns

Offense type	Date
tresspass	2015-01-28 09:30:00
larceny-theft	2015-02-21 08:24:21

Elementwise Functions on Columns

Example:

- use a function to extract a subset of a date (e.g. year, hour) from a column with type = "timestamp"

In [4]:
```
%%sql
SELECT "Date Reported", date_part('hour', "Date Reported")
FROM seattlecrimeincidents
LIMIT 5;
```

```
5 rows affected.
```

Out[4]:

Date Reported	date_part
2015-01-26 13:25:00	13.0
2015-01-29 14:32:00	14.0
2015-01-22 04:35:00	4.0
2015-01-17 01:21:00	1.0
2015-02-02 06:48:00	6.0

Aggregate Functions on Columns

- examples: SUM(), MAX(), MIN(), AVG(), COUNT(), STDDEV()

Data Analysis:

- databases have powerful methods for analyzing data
- one of the most common tasks: applying statistics across groups
- to accomplish this we need to learn
  - how to GROUP sets of data
  - how to apply statistical functions to those groups

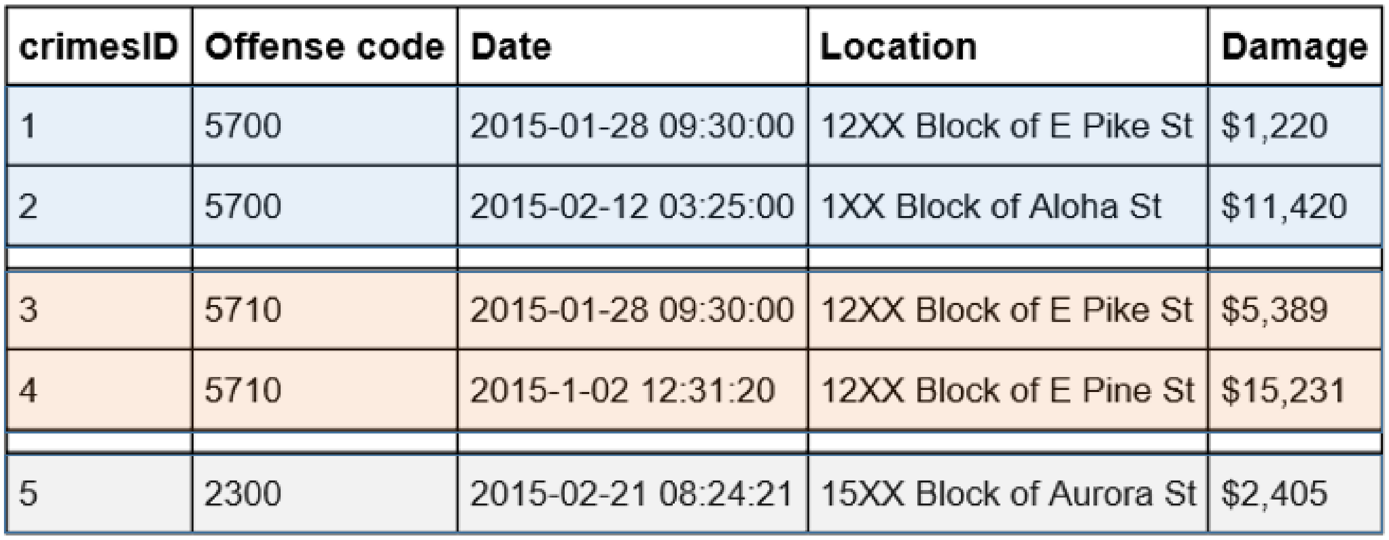
crimesID	Offense code	Date	Location	Damage
1	5700	2015-01-28 09:30:00	12XX Block of E Pike St	$1,220
1	5700	2015-02-12 03:25:00	1XX Block of Aloha St	$11,420
2	5710	2015-01-28 09:30:00	12XX Block of E Pike St	$5,389
2	5710	2015-1-02 12:31:20	12XX Block of E Pine St	$15,231
3	2300	2015-02-21 08:24:21	15XX Block of Aurora St	$2,405

Q: *What is the total damage that occurred for each offense type?*

- data grouped by "Offense code":

crimesID	Offense code	Date	Location	Damage
1	5700	2015-01-28 09:30:00	12XX Block of E Pike St	$1,220
2	5700	2015-02-12 03:25:00	1XX Block of Aloha St	$11,420
3	5710	2015-01-28 09:30:00	12XX Block of E Pike St	$5,389
4	5710	2015-1-02 12:31:20	12XX Block of E Pine St	$15,231
5	2300	2015-02-21 08:24:21	15XX Block of Aurora St	$2,405

- data grouped by "Offense code":



crimesID	Offense code	Date	Location	Damage
1	5700	2015-01-28 09:30:00	12XX Block of E Pike St	$1,220
2	5700	2015-02-12 03:25:00	1XX Block of Aloha St	$11,420
3	5710	2015-01-28 09:30:00	12XX Block of E Pike St	$5,389
4	5710	2015-1-02 12:31:20	12XX Block of E Pine St	$15,231
5	2300	2015-02-21 08:24:21	15XX Block of Aurora St	$2,405

```
SELECT SUM("Damage")
   FROM seattlecrimeincidents
   GROUP BY "Offense code";
```

Offense code	totalDamage
5700	$12,640
5710	$20,620
2300	$2,405

Column Aliasing:

- often we want to rename newly generated columns:

In [5]:
```
%%sql
SELECT "Date Reported", date_part('hour', "Date Reported") AS "reported hour"
FROM seattlecrimeincidents
LIMIT 5;
```

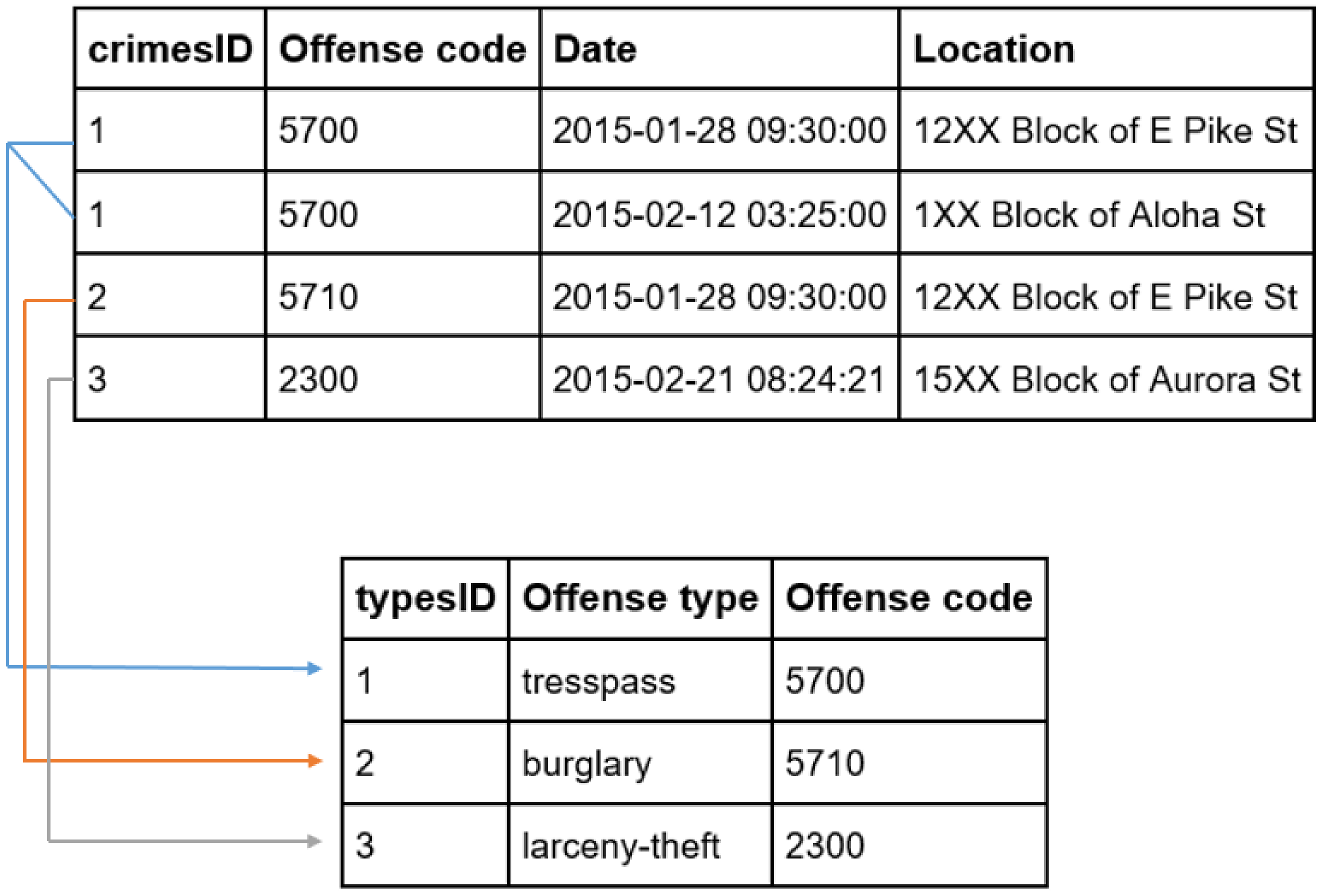
```
5 rows affected.
```

Out[5]:

Date Reported	reported hour
2015-01-26 13:25:00	13.0
2015-01-29 14:32:00	14.0
2015-01-22 04:35:00	4.0
2015-01-17 01:21:00	1.0
2015-02-02 06:48:00	6.0

Joining Tables

- well designed databases distribute data across multiple tables, for efficiency
- then we can JOIN data between tables as needed



crimesID	Offense code	Date	Location
1	5700	2015-01-28 09:30:00	12XX Block of E Pike St
1	5700	2015-02-12 03:25:00	1XX Block of Aloha St
2	5710	2015-01-28 09:30:00	12XX Block of E Pike St
3	2300	2015-02-21 08:24:21	15XX Block of Aurora St

typesID	Offense type	Offense code
1	tresspass	5700
2	burglary	5710
3	larceny-theft	2300

Database Implementation:

- there are many relational database software implementations:
  - commercial: Oracle, Microsoft SQL Server, IBM DB2
  - open source: MySQL, PostgreSQL

- Deployment
  - most databases are deployed on a server
  - can run locally for testing

Database Interface:

- databases are accessed via a *connection string*:
  - hostname, port, user, password
- one can connect through
  - command line
  - GUI apps (pg_admin, MySQL Workbench, DB Visualizer)
  - Python, R, etc.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```