Databases for Data Science

Lecture 02 · 2022-08-31

Course Logistics

Office hours set for 10:00am to 11:30am, Tuesdays and Thursdays

Reminder: Lecture schedule next week

Mon	9/05		No class (Labor Day)
Tue	9/06	4pm	Algorithms
Wed	9/07	5pm	Algorithms
Thu	9/08	4pm	Databases
Fri	9/09	4pm	Databases

Last time

Linux

Using SSH to connect to server

Terminal commands

- `cd`, `pwd`, `ls`
- `nano`, `grep`, `cat`
- `cp`, `mv`, `rm`

Piping and redirection

SQL

Entering the `psql` terminal

Databases, tables, rows, columns

Basic syntax:

```
select ... from ... where ...;
```

Joins, functions, aggregation

Temporary tables

Today's agenda

- **Review basic SQL syntax**
- Inner and outer joins
- Subqueries
- Data types
- File-based workflows

Postgres meta-commands

`\l`: List databases
`\d`: List the tables in the current database
`\d my_table`: Describe the columns in table `my_table`

Basic SQL queries

```
SELECT * FROM
person
WHERE name LIKE "Wiley%";
```

...meaning:

Give me every column
of the rows in `person`
in which the `name` *column matches the pattern:*
("Wiley" followed by any number of characters)

Nuances

- Postgres keeps reading until it sees a `;` - don't forget to type this!
- SQL is a *declarative* language. You are telling the database what you want, not how to find it.
- Each command is interpreted "all at once".
  - For example, in the query `SELECT a.soid FROM arrestees a;`, we use the **alias** `a` before declaring it.
- When testing queries in the terminal, you probably want to `LIMIT` the number of results returned.

Functions

Expressions

```
SELECT to_char(enrollmentdate, "YYYY-MM-DD") as enrolled
FROM students
ORDER BY enrolled;
```

Aggregation functions

```
SELECT AVG(income), birth_year
FROM tax_records
GROUP BY birth_year;
```

Today's agenda

- Review basic SQL syntax
- **Inner and outer joins**
- Subqueries
- Data types
- File-based workflows

Joins

note: for clarity, we will be using the explicit `JOIN` *syntax rather than the "comma join".*

The `JOIN` operators combine multiple tables `ON` shared information.

- There are several ways to do this!

Inner join

```
SELECT a.bookingnumber, charge, agency
FROM charges a
JOIN bookings b ON a.bookingnumber=b.bookingnumber;
```

combine the rows in `a` and `b`
with matching bookingnumbers
and give me:

- *the bookingnumber from `a`,*
- *the charge (from `a`), and*
- *the agency (from `b`).*

Inner join

`person`

user_id	name
1	Alex
1	Avery
2	Blake

`residence`

user_id	city
1	Sarasota
1	Tampa
3	Orlando

What happens if we `JOIN` them `ON a.user_id=b.user_id`?

Inner join

person

user_id	name
1	Alex
1	Avery
2	Blake

INNER JOIN

user_id	name	user_id	city
1	Alex	1	Sarasota
1	Alex	1	Tampa
1	Avery	1	Sarasota
1	Avery	1	Tampa

residence

user_id	city
1	Sarasota
1	Tampa
3	Orlando

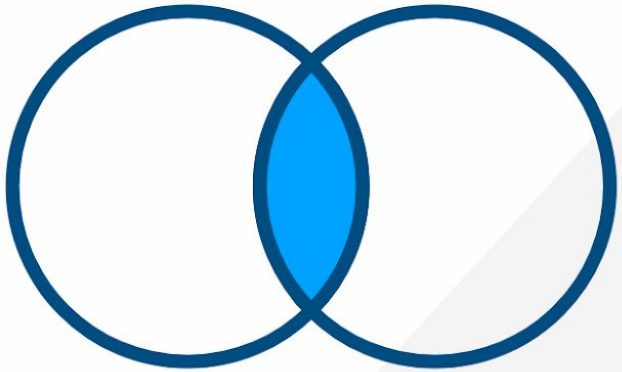
Outer joins

What if we want to include the `person`s without a `residence`?
Or vice versa?

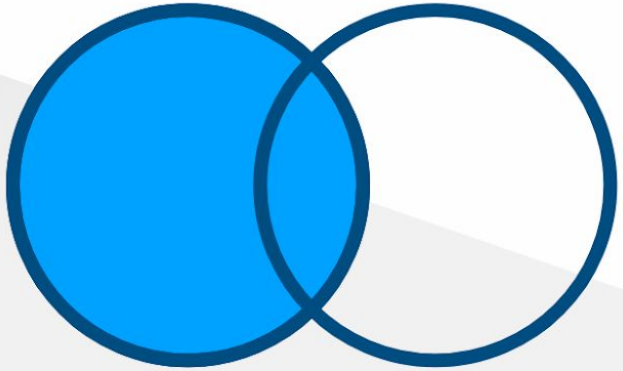
We can use an *outer join* to do this.

Join types

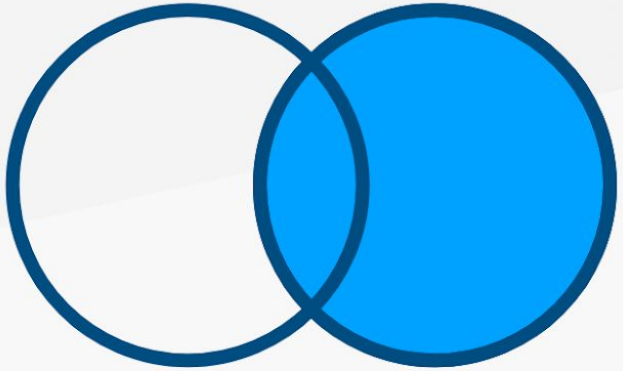
INNER JOIN



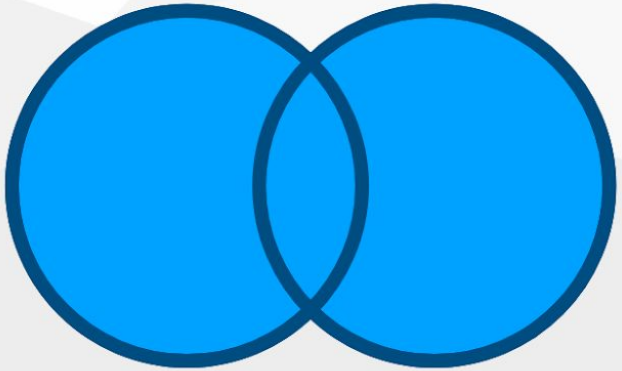
LEFT OUTER JOIN



RIGHT OUTER JOIN



FULL OUTER JOIN



Outer joins

`LEFT JOIN`

	name		city
1	Alex	1	Sarasota
1	Alex	1	Tampa
1	Avery	1	Sarasota
1	Avery	1	Tampa
2	Blake		

`FULL JOIN`

	name		city
1	Alex	1	Sarasota
1	Alex	1	Tampa
1	Avery	1	Sarasota
1	Avery	1	Tampa
2	Blake		
		3	Orlando

`RIGHT JOIN`

	name		city
1	Alex	1	Sarasota
1	Alex	1	Tampa
1	Avery	1	Sarasota
1	Avery	1	Tampa
		3	Orlando

Cross join

Cross product: *all* pairs of rows from both tables.

```
SELECT name, city
FROM person
CROSS JOIN residence;
```

Notice that there is no `ON` clause.

- This could be a lot of stuff!

name	city
Alex	Sarasota
Alex	Tampa
Alex	Orlando
Avery	Sarasota
Avery	Tampa
Avery	Orlando

Today's agenda

- Review basic SQL syntax
- Inner and outer joins
- **Subqueries**
- Data types
- File-based workflows

Subqueries

Running a query produces a result table.

We can run further queries on the results!

Subqueries

A trivial example:

```
SELECT records.name
FROM (
    SELECT *
    FROM person
) AS records;
```

But this has many uses!

Example

Something we tried to do last time:

1. Find the oldest arrestee releasted in each year, and
2. identify the charge for which they were arrested.

Step 1a: find the earliest DOB released in each year.

The relevant columns are `releasedate` and `dob`.

Suggestions?

Step 1a: find the earliest DOB released in each year.

```
SELECT
    to_char(releasedate, 'YYYY') AS year,
     min(dob) AS oldest
FROM bookings
GROUP BY year
ORDER BY year;
```

Step 1b: find the oldest arrestee released in each year.

Consider the approach below:

```
SELECT
    soid,
    to_char(releasedate, 'YYYY') AS year,
    min(dob) AS oldest
FROM bookings
GROUP BY year
ORDER BY year;
```

Why doesn't this work?

Solution using an inner query

```
SELECT DISTINCT
    soid,year,oldest
FROM (
    SELECT
    to_char(releasedate, 'YYYY') AS year,
    min(dob) AS oldest
    FROM bookings
    GROUP BY year
) a JOIN bookings b ON a.oldest=b.dob AND a.year=to_char(b.releasedate, 'YYYY')
ORDER BY year;
```

Today's agenda

- Review basic SQL syntax
- Inner queries
- Inner and outer joins
- **Data types**
- File-based workflows

Data types

Columns in SQL databases are strongly typed.

- `boolean`
- `int`, `numeric`
- `char(n)`, `varchar(n)`, `text`
- `timestamp`, `date`, `time`, `interval`

There are many, many more.

Data types

Columns in SQL databases are strongly typed.

- `boolean`
- `int`, `numeric`
- `char(n)`, `varchar(n)`, `text`
- `timestamp`, `date`, `time`, `interval`

There are many, many more.

Null

What happens when a column is empty?

- When might we see this, even when using a complete dataset?

Today's agenda

- Review basic SQL syntax
- Inner queries
- Inner and outer joins
- Data types
- **File-based workflows**

Running SQL from files

You can run SQL from files as follows:

```
[user@cs1 ~]$ psql -a -d DBNAME -f FILENAME
```

```
DBNAME=# \i FILENAME
```

Writing results to files

```
[user@cs1 ~]$ psql -a -d DBNAME -f INPUT_FILE -o OUTPUT_FILE
```

Set the output of the psql terminal:

```
DBNAME=# \o OUTPUT_FILE
```

Revert to console output

```
DBNAME=# \o
```

CSV output

```
[user@cs1 ~]$ psql -a -d DBNAME -f INPUT_FILE -o OUTPUT_FILE --csv
```

Or, inside the psql terminal:

```
\copy (select * from person) to OUTPUT_FILE with csv delimiter ',' header
```

Upcoming lectures

#	Date	Topics
01	08-29	• Linux • Relational DB concepts • SQL basics
02	**08-31**	**• More query syntax • Data types • Files**
03	09-08	• SQL create, update, delete • Transactions • Schemas and UML • Normal forms
04	09-09	• Indexes • Query plans • Stored procedures
05	09-13	• Running SQL from Python • numpy

Other notes

- First homework to be released soon
- Also a brief survey quiz (required, but not graded)
- Will make a Canvas announcement about VPN access