Foundations of Data Science for Biologists

Introduction to SQL

BIO 724D

26-FEB-2024

Instructors: Greg Wray and Paul Magwene

Introduction to SQL

What is SQL?

SQL is a language for interacting with tabular data

- Specifically, around the idea of relational data structures

SQL is designed:

- for **powerful data queries** using a simple and compact syntax
- to enforce **data integrity** during data entry and updating
- for **highly efficient** search, sort, summarize, etc. operations
- to be **massively scalable** (billions of rows, thousands of columns)

SQL is a **domain-specific language** (DSL)

- Intended for a specific set of needs
- Best-in-class for its intended purposes, bad-to-terrible for most other tasks

SQL is designed to work with relational databases

observations

seq	genus	species	date	location
1022	Colias	striatus	2007-06-14	Fairview Hotel
1023	Pycnonotus	tricolor	2007-06-14	Fairview Hotel
1024	Milvus	migrans	2007-06-14	Fairview Hotel
1032	Chalcomitra	amethystina	2007-06-14	Sheldrick Centre
1033	Pycnonotus	tricolor	2007-06-14	Sheldrick Centre
1050	Lamprotornis	superbus	2007-06-15	Lake Naivasha
1051	Lamprotornis	purpuroptera	2007-06-15	Lake Naivasha
1052	Scopus	umbretta	2007-06-15	Lake Naivasha
1053	Buteo	augur	2007-06-15	Hell's Gate NP
1054	Cisticola	marginata	2007-06-15	Hell's Gate NP

species

seq	genus	species	ssp	IUCN	familiar
1418	Pynonotus	leucogenys	1	LC	Himalayan Bulbul
1419	Pynonotus	barbatus	5	LC	Common Bulbul
1420	Pynonotus	tricolor	3	LC	Dark-capped Bulbul
1421	Pynonotus	capensis	1	LC	Cape Bulbul

locations

location	prov	country	clim	elev	geolocation
Everard Reserve	Vic	Australia	Csb	90	-37.68,145.49
Everglades NP	FL	USA	Aw	2	25.39,-80.63
Fairview Hotel	Nb	Kenya	Cfb	1715	-1.29,36.80
Faskrudsfjordur	Au	Iceland	ET	20	64.93,-14.01

Relational design removes redundant information by spreading data across tables:

(1) reduces errors, (2) simplifies updates, (3) saves space, (4) speeds up queries

Important points about how relational databases work

Every table contains a **primary key** consisting of 1 row (or, occasionally, more)

- Value(s) are required (no blanks or NULL values)
- Values must be unique (thus, no duplicate rows are possible)

Every table should contain at least one **relation**

- A column intended to reference a column in another table
- All this requires is that they contain the same data type (optionally, more restrictive)

The order of rows in a table is not consistent and is not stable

- This allows for optimized query and sort operations
- However, guaranteed sort order must be explicitly defined

SQL is limited to a small set of operations

Only four kinds of operations are permitted with relational databases

Create, **R**ead, **U**pdate, **D**elete (CRUD)

`CREATE` to create a new database, table, or relation; `INSERT` to add rows to a table

`SELECT` to query a database

`UPDATE` to change the information within a table

`DELETE` to remove rows from a table; `DROP` to remove a database, table, or relation

SQL has some additional keywords, but most work with the above set

Clauses within statements: `WHERE`, `JOIN`, `GROUP BY`, `LIMIT`, etc.

Operators within statements: `=`, `>`, `IN`, `NOT`, `LIKE`, `BETWEEN`, etc.

Functions within statements: `MIN`, `MEAN`, `COUNT`, `DISTINCT`, etc.

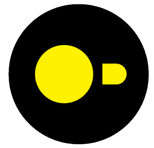
`Pandas` and `dplyr` implement many SQL-like operations (and were inspired by it)

Getting started

Software

We will use **DuckDB**, one of many implementations of the SQL standard

- Free, open source, runs on all major platforms, straightforward to implement
- Simple: no external software dependencies, no need to set up a RDBMS server
- Flexible: can be accessed directly from any Python program



DuckDB

In addition, we will use **JupySQL** to interface DuckDB with Jupyter

- Also free, open source, etc.

JupySQL

There are other ways to access SQL from Python

- **sqlite3**: a Python standard library for SQLite in widespread use
- **psychoPG3**: a Jupyter extension for PostgreSQL
- **SQLalchemy**: a highly flexible general-purpose interface

Points to keep in mind as we construct queries

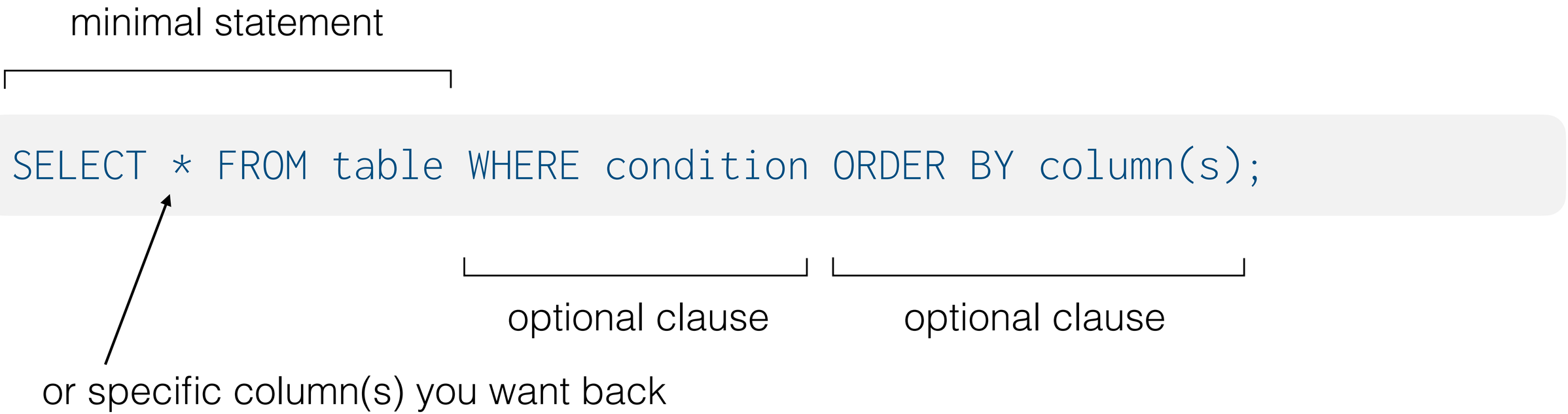
SQL is case-insensitive, but there are conventions

- SQL keywords and functions are in ALL CAPS
- Table and column names (“identifiers”) are in lowercase

Statements (lines of code) end with a semicolon

- Required in scripts
- Not required in single-line queries, but good practice

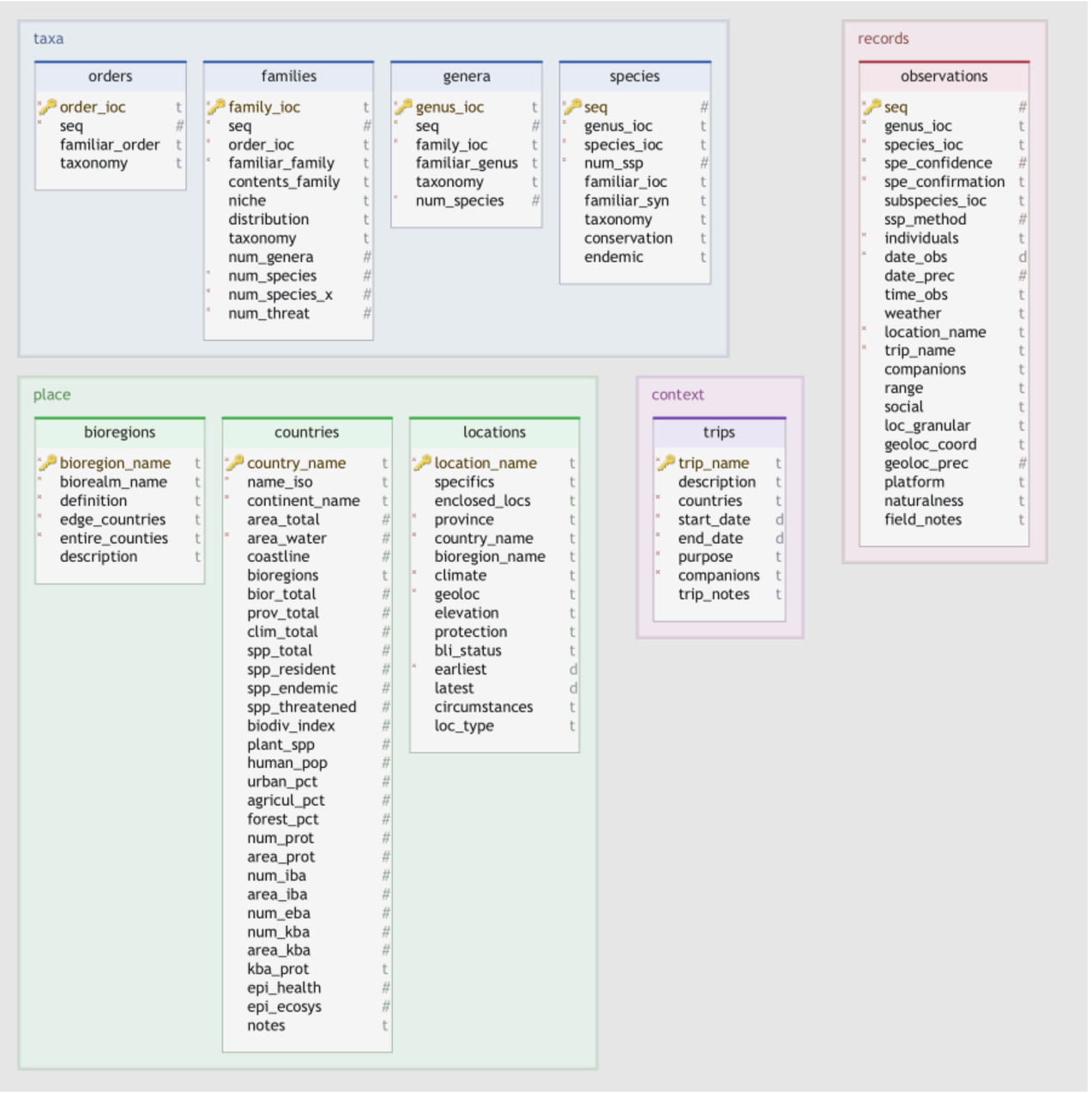
SELECT syntax



Clause order matters: `FROM`, `JOIN / ON`, `WHERE`, `GROUP BY`, `ORDER BY`, `LIMIT`

Practice database

Practice database: field observations of birds



Full implementation consists of 9 tables

- 1 records field observations
- 4 define nested taxonomic groups
- 3 define place
- 1 defines context

Simplified version omits 2 place tables

And some columns from every table

Points to keep in mind about the database

Taxonomy follows the International Ornithological Congress (IOC)

- Scientific (or Latin) taxon name: single word, based on rules of zoological nomenclature
- Familiar (or common) name: single name, an attempt to reconcile multiple synonyms
- Sequence of taxa within each rank: provides consistent ordering in lists

The **content** of taxonomic tables differ by rank

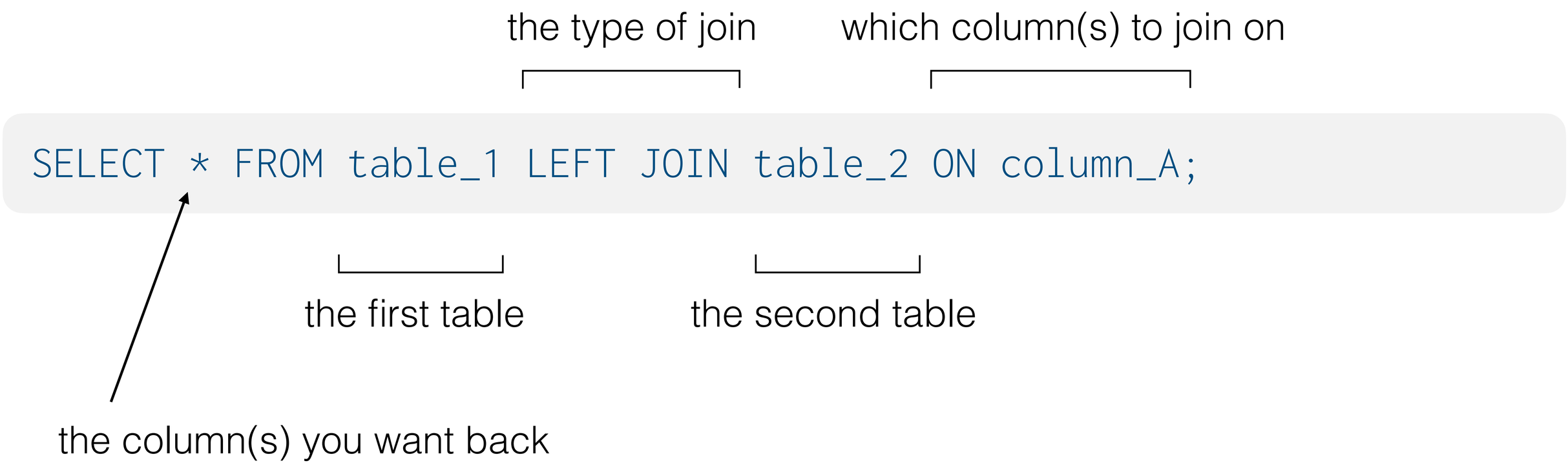
- `orders` and `families`: complete list of taxa recognized by the IOC
- `genera` and `species` : only taxa that have been observed
- `subspecies` : not in a separate table but individually recorded under observations

The **primary keys** of taxonomic tables differ by rank

- `orders`, `families`, and `genera`: IOC scientific name
- `species` : IOC sequence (the binomen introduces complexity!)
- `subspecies` : not relevant

Joins

Recommended join syntax



There are other ways to specify joins, some of which are short-cuts

This syntax is recommended as the most readable and specific

Joins append rows from one table with those of another according to rules

Left join: keep every row from table 1; append matching rows from table 2

```
SELECT * FROM observations LEFT JOIN locations ON loc_name;
```

observations

seq	genus	species	date	loc_name
1022	Colias	striatus	2007-06-14	Fairview Hotel
1023	Pycnonotus	tricolor	2007-06-14	Fairview Hotel
1024	Milvus	migrans	2007-06-14	Fairview Hotel
1032	Chalcomitra	amethystina	2007-06-14	Sheldrick Centre
1033	Pycnonotus	tricolor	2007-06-14	Sheldrick Centre
1050	Lamprotornis	superbus	2007-06-15	Lake Naivasha
1051	Lamprotornis	purpuroptera	2007-06-15	Lake Naivasha
1052	Scopus	umbretta	2007-06-15	Lake Naivasha
1053	Buteo	augur	2007-06-15	Hells Gate NP
1054	Cisticola	marginata	2007-06-15	Hells Gate NP

locations

loc_name	prov	country	clim	elev	geolocation
Everard Reserve	Vic	Australia	Csb	90	-37.68,145.49
Everglades NP	FL	USA	Aw	2	25.39,-80.63
Fairview Hotel	Nb	Kenya	Cfb	1715	-1.29,36.80
Faskrudsfjordur	Au	Iceland	ET	20	64.93,-14.01
Hell’s Gate NP	RV	Kenya	Cfb	1800	-0.89,36.32
Hinxton	Cam	England	Cfb	40	52.08,0.17
Lago Huacarpay	Cuz	Peru	Cwb	3800	-13.61,-71.72
Lake Naivasha	RV	Kenya	Cfb	1700	-0.82,36.38
Sheldrick Centre	Nb	Kenya	Cfb	1750	-1.33,36.77
Siem Reap	SiR	Cambodia	Aw	20	13.36,103.86

Joins append rows from one table with those of another according to rules

Left join: keep every row from table 1; append matching rows from table 2

```
SELECT * FROM observations LEFT JOIN locations ON loc_name;
```

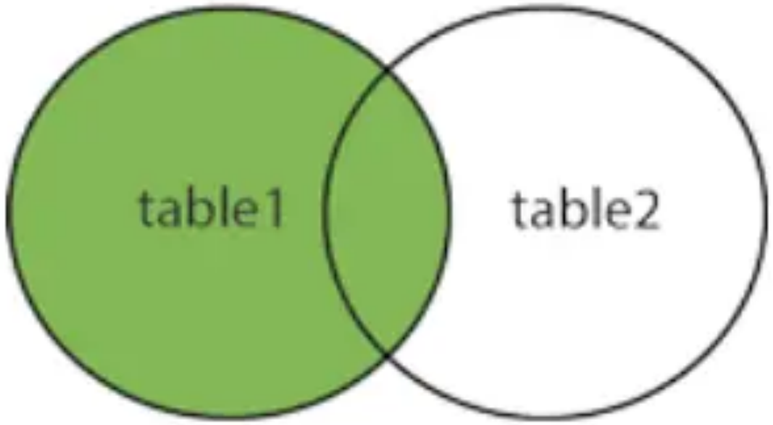
return

seq	genus	species	date	location	prov	country	clim		
1022	Colias	striatus	2007-06-14	Fairview Hotel	Nb	Kenya	Cfb	1715	-1.29,36.80
1023	Pycnonotus	tricolor	2007-06-14	Fairview Hotel	Nb	Kenya	Cfb	1715	-1.29,36.80
1024	Milvus	migrans	2007-06-14	Fairview Hotel	Nb	Kenya	Cfb	1715	-1.29,36.80
1032	Chalcomitra	amethystina	2007-06-14	Sheldrick Centre	Nb	Kenya	Cfb	1750	-1.33,36.77
1033	Pycnonotus	tricolor	2007-06-14	Sheldrick Centre	Nb	Kenya	Cfb	1750	-1.33,36.77
1050	Lamprotornis	superbus	2007-06-15	Lake Naivasha	Nb	Kenya	Cfb	1750	-1.33,36.77
1051	Lamprotornis	purpuroptera	2007-06-15	Lake Naivasha	Nb	Kenya	Cfb	1750	-1.33,36.77
1052	Scopus	umbretta	2007-06-15	Lake Naivasha	Nb	Kenya	Cfb	1750	-1.33,36.77
1053	Buteo	augur	2007-06-15	Hells Gate NP	RV	Kenya	Cfb	1800	-0.89,36.32
1054	Cisticola	marginata	2007-06-15	Hells Gate NP	RV	Kenya	Cfb	1800	-0.89,36.32

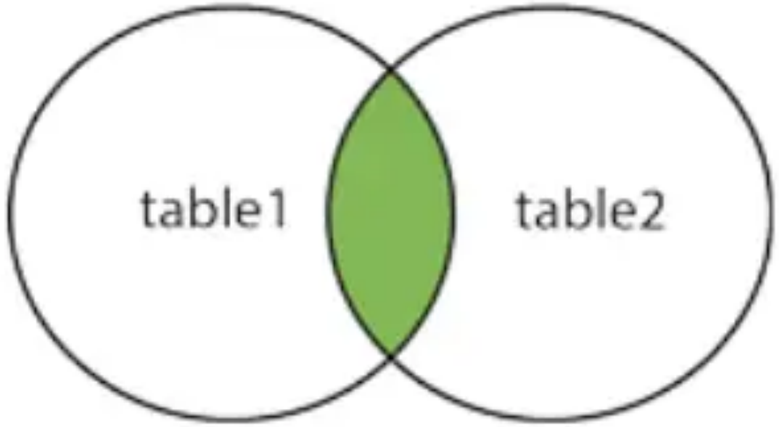
Note that there is a one-to-many relationship between observations and locations

Joins can return different sets of rows

Left (outer) join



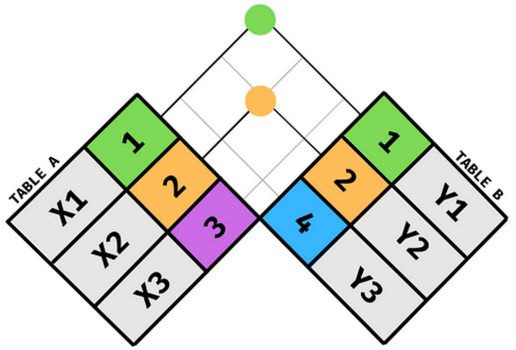
Inner join



Inner joins only return rows where there is a match in both tables

Left joins return every row of table 1 and will insert NULL where there is no match in table 2

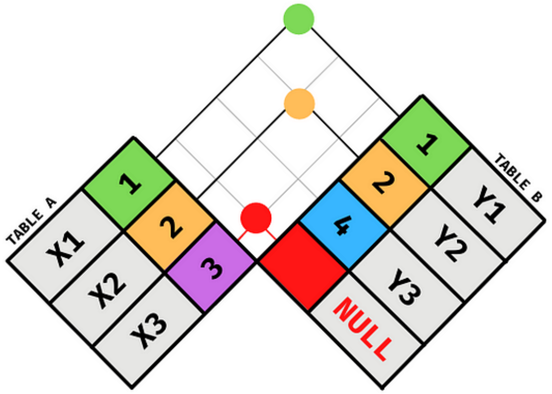
Joins can return different sets of rows



INNER JOIN

```
SELECT
    <SELECT LIST>
FROM       TABLE_A A
INNER JOIN TABLE_B B
    ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2



LEFT JOIN

```
SELECT
    <SELECT LIST>
FROM      TABLE_A A
LEFT JOIN TABLE_B B
    ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2
3	X3	NULL

Inner joins only return rows where there is a match in both tables

Left joins return every row of table 1 and will insert NULL where there is no match in table 2

Congratulations! You just learned the most difficult kind of query in SQL