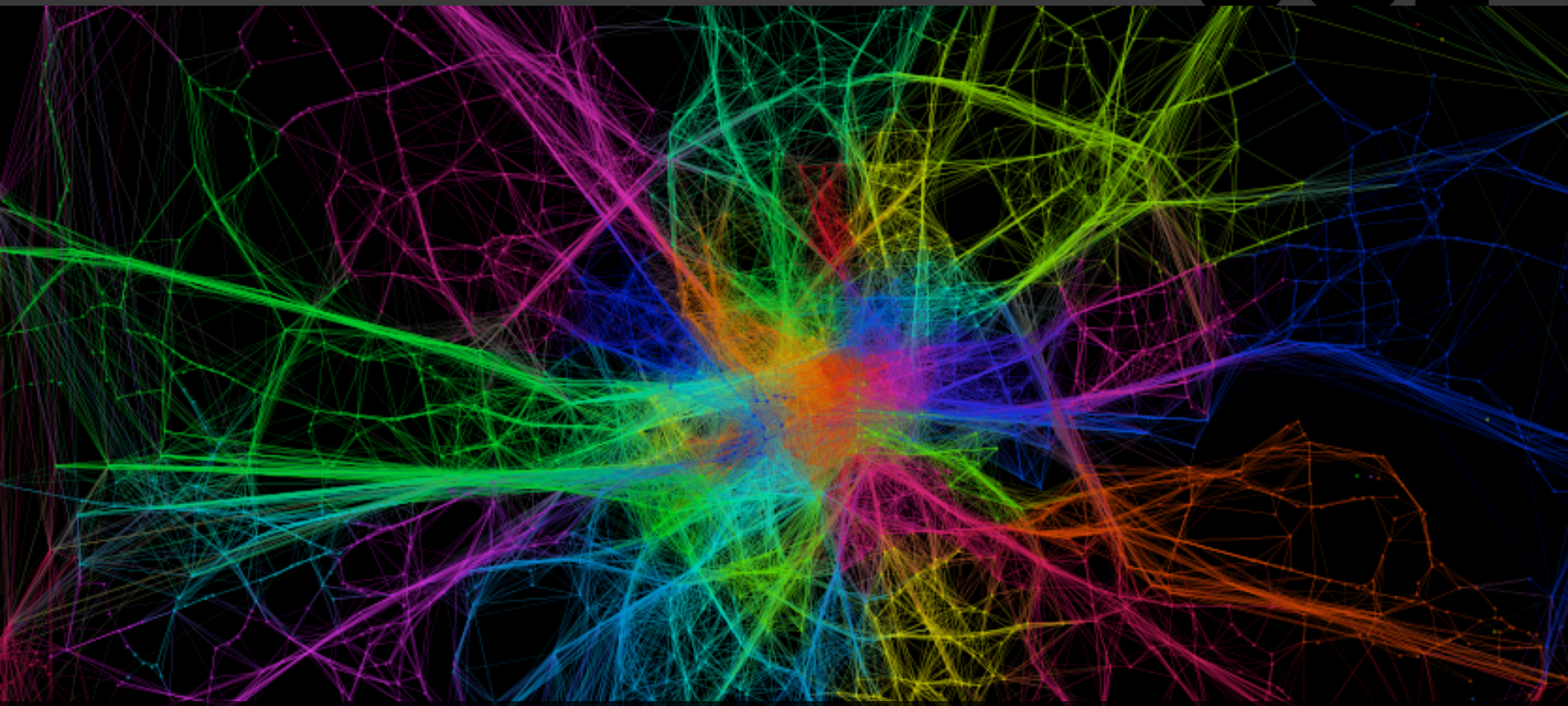
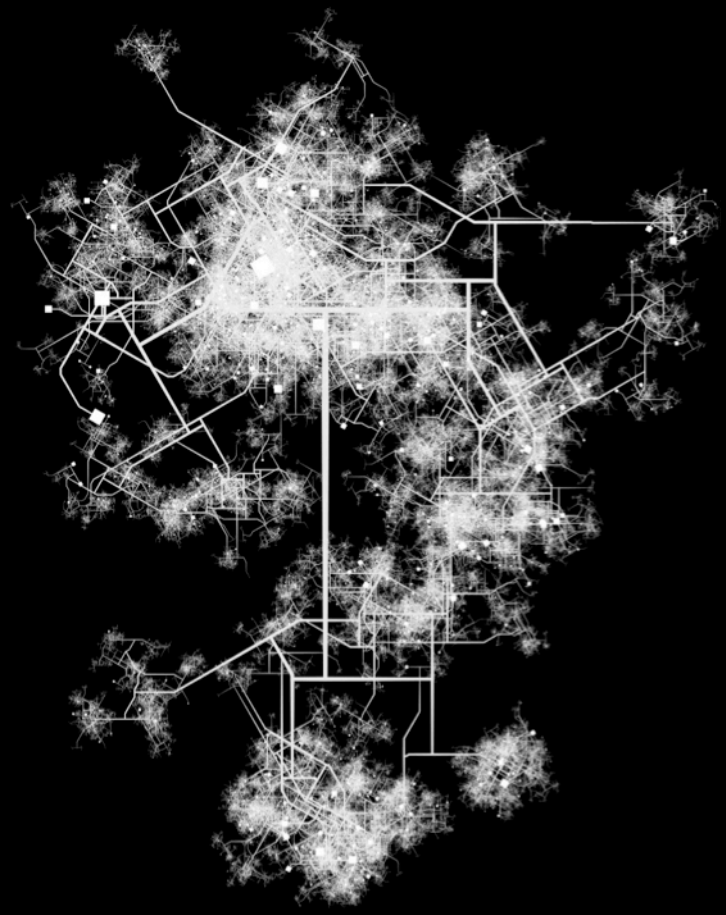


Introduction to SQL

CASA0006 : Data Science for Spatial Systems
CASA0009: Spatial Data Capture and Analysis

Steven Gray

Outline



1. What is SQL?
2. Querying Data
   a. Refining Queries
   b. Query Syntax
3. Organising Results
   a. Aggregation
   b. Ordering
   c. Limiting
4. Amending Data
   a. Inserts
   b. Updates
   c. Deletes

What is SQL?

Recap

- Allows you to talk to your database
  - Let's you get data
  - Let's you add data
  - Let's you sort data
  - Let's you amend data
  - Let's you run calculations and summary statistics
- Turns your raw rows and columns into meaningful information
- It's a relatively straightforward language, well-structured with limited syntax
- Cross-platform and universal standard, although there are different flavours

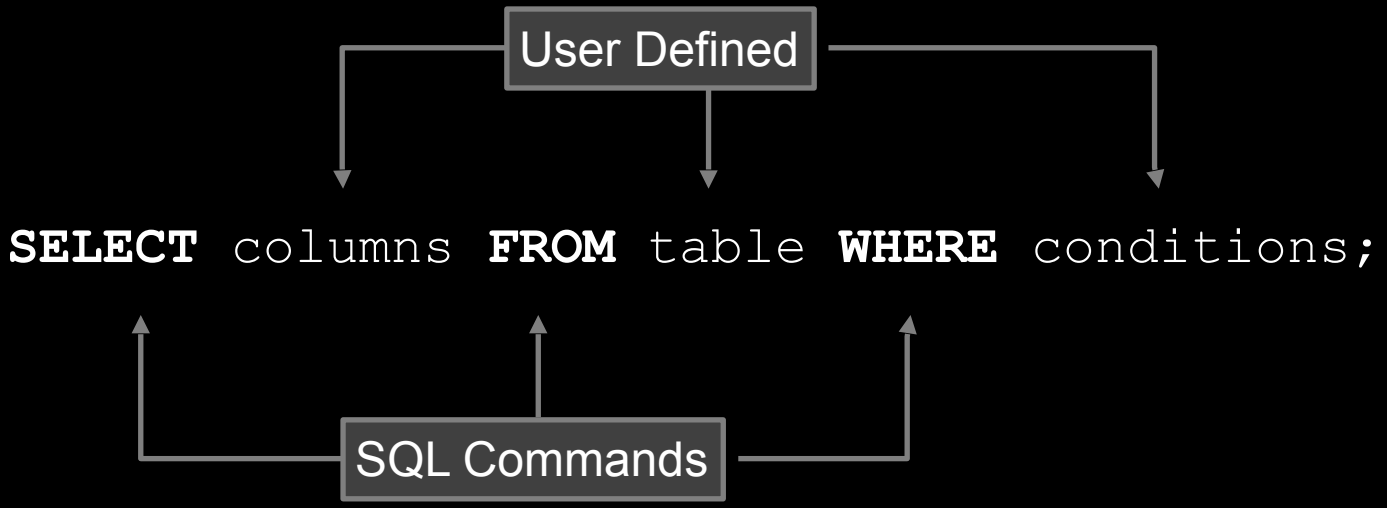
Querying Data

Building Queries
Query Logic

Building Queries

The Main Thing

Most of your database work can be completed with one simple statement



SELECT … WHERE

Querying Data

Querying allows you to select which **columns** and **records** you wish to see

```
SELECT * FROM cities;
```

All columns, all records

```
SELECT name, pop FROM cities;
```

Two columns, all records

```
SELECT name, pop FROM cities
WHERE country_cd = 'GB';
```

Two columns, some records

One condition on text column

```
SELECT DISTINCT name FROM cities;
```

One column, some records
No duplicate names

SELECT … WHERE

Extending Conditions

Conditions can be combined to produce highly specific results

```
SELECT * FROM cities
WHERE pop >= 1000000 AND elev < 100
AND capital = true;
```

Combines conditions

Numeric value

Boolean value

```
SELECT * FROM cities
WHERE pop >= 100000 AND elev > 500 AND elev < 1000
AND (capital = true OR category = 'Major');
```

Same column twice

SELECT … WHERE

Advanced Conditions

Advanced syntax is required for some data types and contexts

```
SELECT name FROM cities WHERE name LIKE 'San %' ;
```

LIKE used for similarity

% represents partial match
_ represents a single char

```
SELECT name FROM cities
WHERE pop BETWEEN 1000000 AND 5000000;

SELECT * FROM cities
WHERE name IN ('Paris','London');
```

SELECT … WHERE

Column Arithmetic

New columns can be created through simple **arithmetic** and **functions**

```
SELECT name, pop/count_pop AS prop FROM cities;
```

Simple arithmetic on two values

Alias

```
SELECT AVG(pop) FROM cities;
```
Mean population

```
SELECT COUNT(*) FROM cities;
```
Count of all records

```
SELECT MAX(pop) FROM cities;
```
Maximum population

```
SELECT MIN(pop) FROM cities;
```
Minimum population

```
SELECT SUM(pop) FROM cities;
```
Sum population

SELECT … WHERE

For Reference: SQL Operators and Functions

Operators – Combine conditions or values

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
=	Equals
!= or <>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Operator	Description
AND	Combines multiple conditions
OR	Combines multiple conditions
BETWEEN	Only values between
EXISTS	Searches for presence of row meeting criteria
IN	Finds values within provided list
LIKE	Finds values similar to provided value
NOT	Negates other operators (e.g. NOT BETWEEN, NOT IN)
IS NULL	Returns if value is null

Functions – Return values calculated from values within named column (*col*)

Function	Description
AVG(*col*)	Returns average of all values in column
COUNT(*col*)	Returns count of all rows in column
FIRST(*col*) *and* LAST(*col*)	Returns first or last value from all values in column
MIN(*col*) *and* MAX(*col*)	Returns maximum or minimum value from all values in column
SUM(*col*)	Returns sum of all values in range
LEN(*col*)	Returns length of a text value
ROUND(*col, val*)	Rounds numeric value to given number of decimals (*val*)
UCASE(*col*)	Coverts text field to upper case
LCASE(*col*)	Coverts text field to lower case

SELECT … WHERE

Subqueries

Subqueries allow you to **SELECT** from subsets of the data

City's country code

Country code

```
SELECT * FROM cities
WHERE country_cd IN (SELECT country_cd FROM countries
WHERE pop > 100000000);
```

Country population

Returns list and we check if value is IN the list

```
SELECT * FROM cities
WHERE pop > (SELECT MIN(pop) FROM countries);
```

Returns single value and compares values

Organising Results

Aggregation
Ordering
Limiting


FIRE ENGINE

GROUP BY

Aggregation

Allows us to group records together based on **shared attributes**, and calculate statistics for each group

```
SELECT country_cd, COUNT(*), AVG(pop) FROM cities
GROUP BY country_cd;
```

Returns number of cities and mean urban population for each country

```
SELECT country_cd, COUNT(*), AVG(pop) FROM cities
WHERE continent = 'EU'
GROUP BY country_cd;
```

Restricted to Europe only

HAVING

Aggregation

Annoyingly, aggregates statistics can not be used within **WHERE** statements, so the **HAVING** statement was created

```
SELECT country_cd, COUNT(*), AVG(pop) FROM cities
WHERE continent = 'EU'
GROUP BY country_cd
HAVING AVG(pop) >= 100000;
```

Limits results to only cities with mean urban population of over 100000

```
SELECT continent, AVG(pop) FROM cities
GROUP BY continent
HAVING AVG(pop) >= 50.0;
```

ORDER BY

Sorting

This orders the returned set of records by a specified attribute

```
SELECT name, pop FROM cities
ORDER BY pop DESC;
```

Returns city name and population in order of population

```
SELECT name, pop FROM cities
ORDER BY pop ASC;
```

```
SELECT name, pop FROM cities
ORDER BY pop, name;
```

```
SELECT country_cd, COUNT(*) AS city_count, SUM(pop)
AS urban_pop FROM cities
GROUP BY country_cd
ORDER BY city_count DESC;
```

Ordering applied to grouped summary statistics

Amending Data

Inserts
Updates
Deletes

INSERT INTO

Adding Data

At some point you'll need to add new rows to your table, for this you use **INSERT INTO**

Table name

```
INSERT INTO cities
VALUES (99999,'Grantham','Grantham',52.918,0.638,
'GB','EU',41998,63,'Europe/London',false,'Minor');
```

VARCHAR DOUBLE INT BOOLEAN

Attributes must align with **order** and **data types** within table definitions

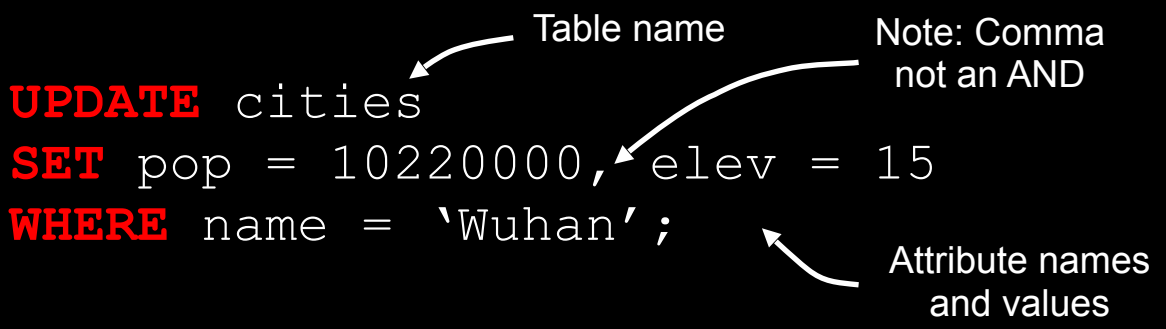
```
INSERT INTO cities (id,name,lat,lon)
VALUES (99999,'Grantham',52.918, 0.638);
```

Adding only selected attributes, others remain NULL

UPDATE

Amending Data

This command allows you to alter all or some **existing records**



Updates specified attributes where a condition is met

```
UPDATE cities
SET continent = 'EU';
```

!WARNING!
No condition = **ALL** records updated!

DELETE

Removing Data

This is how you **remove existing records** from a table

Table name

```
DELETE FROM cities
WHERE city = 'Wuhan';
```

Removes a record where a condition or conditions are met

```
DELETE FROM cities
WHERE continent = 'AS' and pop < 3000000;
```

```
DELETE FROM cities;
TRUNCATE TABLE cities;
```

!WARNING!
Deletes all records!

Why SQL?

versus Pandas or R data handling tools

- SQL is **much more efficient** at handling much of the grunt work involved in working with data
- Allows quick extraction of subsets of large datasets, without causing lag to a client machine
- Highly efficient for **creating, editing, updating and deleting** records and columns, and very quick at generating summary statistics
- Enables efficient **joins** across multiple datasets
- Often a **simpler syntax** to do all of the above
- In many real-world contexts, you'll be working in environments where you draw from a shared database

Questions?

Steven Gray

steven.gray@ucl.ac.uk

@frogo

Workshop

Introduction to SQL

- In this workshop, you'll develop your SQL skills and continue using MySQL Workbench, connecting to the database you setup last week

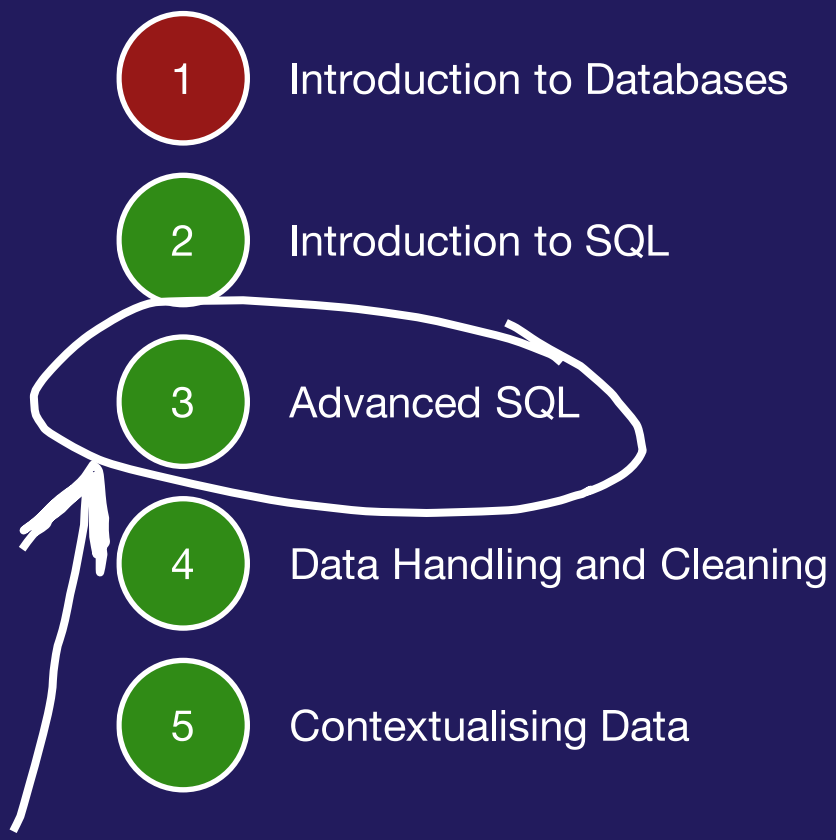
- **Download the Worksheet and Data from Moodle**
- There is also a **quiz** to complete on Moodle

- Start working on this at home – come to the drop in session **if you need help** – and use **Slack AnswerBot** to for hints and answers
- Drop In: Birkbeck Malet Street 414/415, Friday 11am-1pm

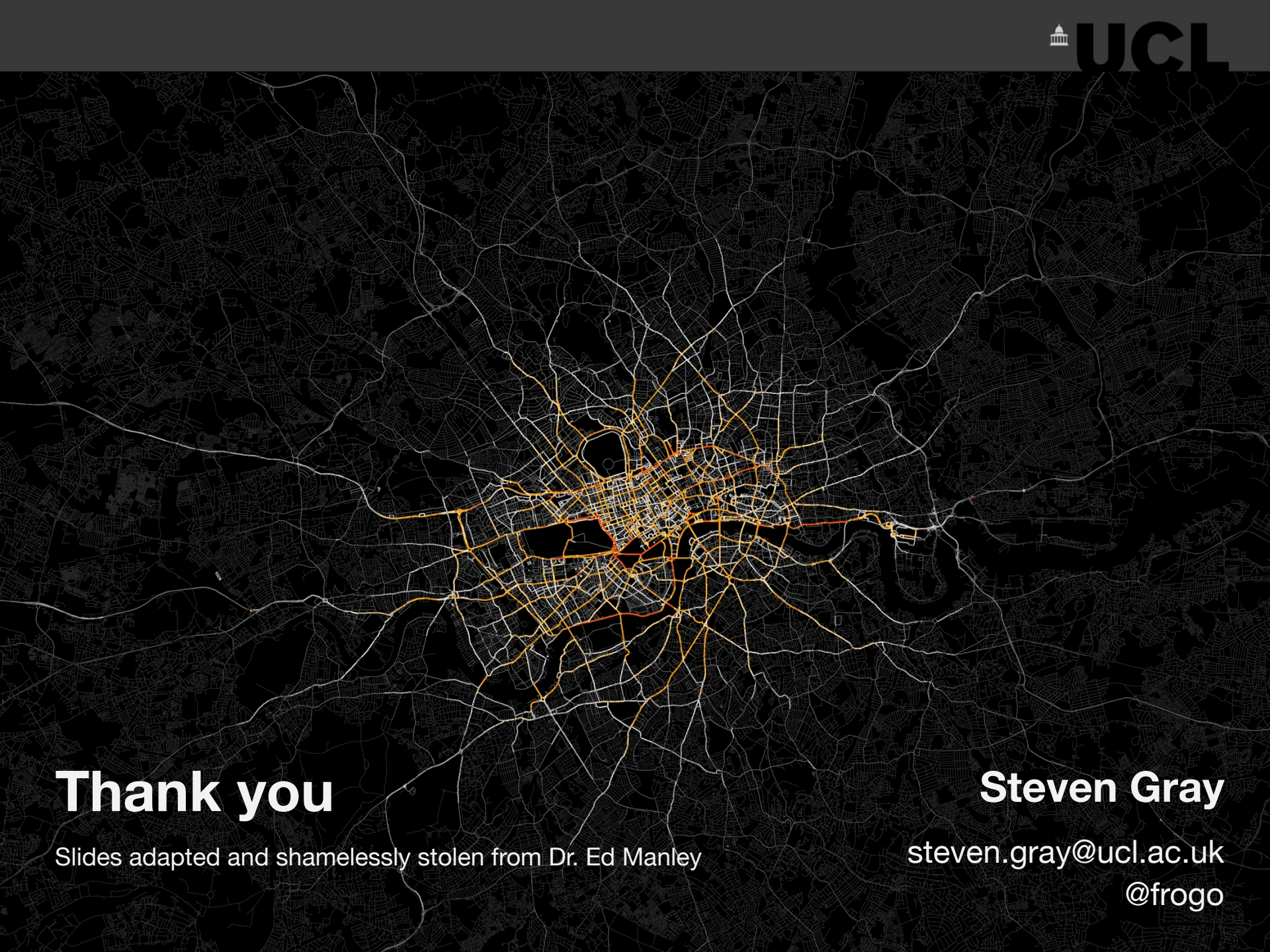
Spatial Data Capture Coursework

Groups

- If you don't have a group, come to the front NOW
- Groups can only be of five – if you're in a three or four, come to the front and pick up a new team member
- If you're in a group of six, you got to kick someone out, sorry ☹
- Data Science students, you are still on your own…



NEXT WEEK'S LECTURE WILL BE AT 4PM-5PM



Thank you

Slides adapted and shamelessly stolen from Dr. Ed Manley

Steven Gray

steven.gray@ucl.ac.uk

@frogo