SQL for Data Science

Mona Khalil @ ODSC Europe
September 16, 2020

Topics

1. Overview of relational databases
    - SQL Query Syntax
    - Aggregating, filtering, and sorting
2. Combining Data from Multiple Tables
    - Identifying keys in tables
    - Selecting the correct join
    - Mathematical calculations on your data
3. Transforming Your Data for Analysis
    - Conditional statements
    - Answering business questions
    - Preparing a dataset for a statistical project

If you haven't done so already

- Go to github.com/mona-kay/odsc-sql-for-data-science to download the `global_powerplants.db` database file
- Go to sqlitetutorial.net/download-install-sqlite/ to set up sqlite3 and SQLite Studio

About

- Data Scientist at Greenhouse Software
  - Teaching SQL for >3 years
  - Intermediate SQL instructor with DataCamp
  - Passionate about analytics, effective communication and research methods use

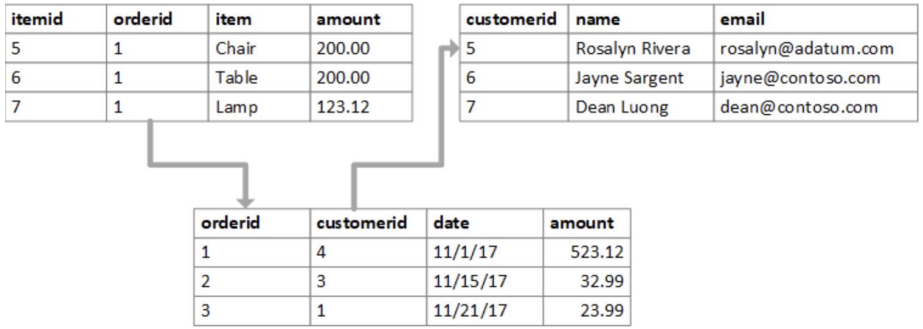
Twitter: mona_kay_

Why SQL?

- **Structured Query Language** (SQL) has been used to retrieve and shape information from relational databases for 40+ years
    - Most commonly requested/utilized skill among data scientists
    - Foundation of data manipulation and analysis in Python and R

Relational Databases 101

- A **relational database** is a set of tables with relationships that determine how the information in each *relates* to other tables
    - Tables are connected by **keys** (columns that identify data across tables)
    - Stores data efficiently and makes it broadly accessible to multiple stakeholders



itemid	orderid	item	amount
5	1	Chair	200.00
6	1	Table	200.00
7	1	Lamp	123.12

customerid	name	email
5	Rosalyn Rivera	rosalyn@adatum.com
6	Jayne Sargent	jayne@contoso.com
7	Dean Luong	dean@contoso.com

orderid	customerid	date	amount
1	4	11/1/17	523.12
2	3	11/15/17	32.99
3	1	11/21/17	23.99

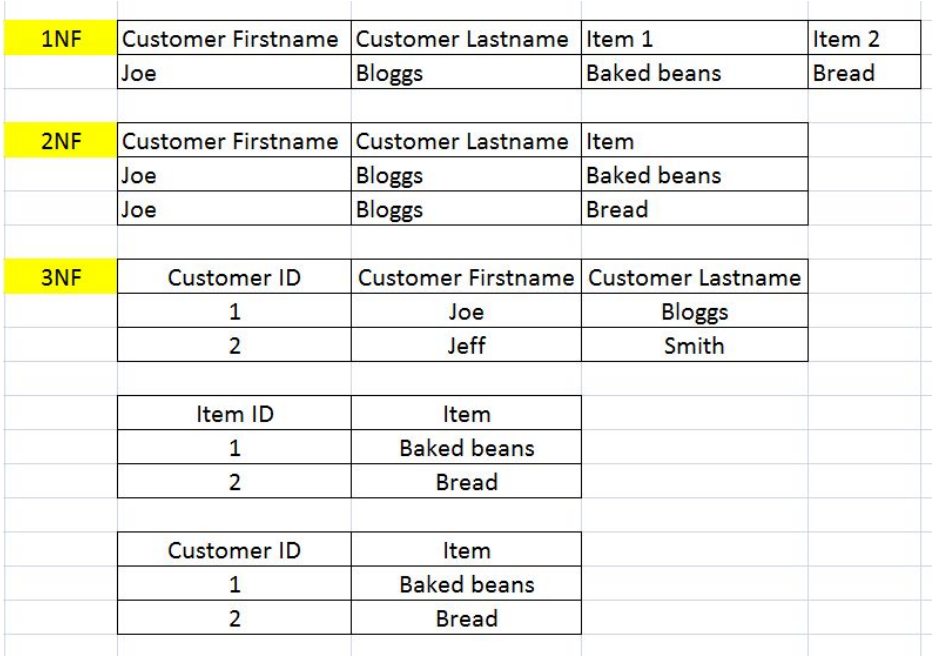
Relational Databases 101

- Data stored in flat files presents numerous problems
  - Storage is on the level of the most granular piece of data
  - Repeat/duplicate information
  - Less than optimal for most questions you'll have from the data

country	country_long	name	gppd_idnr	capacity_mw	latitude	longitude	primary_fuel
AFG	Afghanistan	Kajaki Hydroelectric Power Plant	GEODB0040538	33	32.322	65.119	Hydro
AFG	Afghanistan	Mahipar Hydroelectric Power Plant	GEODB0040541	66	34.556	69.4787	Hydro
AFG	Afghanistan	Naghlu Dam Hydroelectric Power Plant	GEODB0040534	100	34.641	69.717	Hydro
AFG	Afghanistan	Nangarhar (Darunta) Hydroelectric Power Plant	GEODB0040536	11.55	34.4847	70.3633	Hydro
AFG	Afghanistan	Northwest Kabul Power Plant	GEODB0040540	42	34.5638	69.1134	Gas
AFG	Afghanistan	Pul-e-Khumri Hydroelectric Power Plant	GEODB0040537	6	35.9416	68.71	Hydro
AFG	Afghanistan	Sarobi Dam Hydroelectric Power Plant	GEODB0040535	22	34.5865	69.7757	Hydro
ALB	Albania	Bistrica 1	WRI1002169	27	39.9116	20.1047	Hydro
ALB	Albania	Fierza	WRI1002170	500	42.2514	20.0431	Hydro
ALB	Albania	Koman	WRI1002171	600	42.1033	19.8224	Hydro
ALB	Albania	Lanabregas	WRI1002172	5	41.3428	19.8964	Hydro
ALB	Albania	Shkopet	WRI1002173	24	41.6796	19.8305	Hydro
ALB	Albania	Ulez	WRI1002174	25	41.6796	19.8936	Hydro
ALB	Albania	Vau i Dijes	WRI1002175	250	42.0137	19.6359	Hydro
ALB	Albania	Vlora	WRI1002176	98	40.4874	19.434	Other
DZA	Algeria	Ain Djasser	WRI1023776	520	35.8665	6.0262	Gas
DZA	Algeria	Annaba	WRI1023795	71	36.8924	7.7634	Gas

Relational Databases 101

- Data is sorted into relational databases using a process called normalization
  - Limit tables to one topic/purpose
  - Store **records** (pieces of information) as **rows**
  - Remove duplicates and optimize for space
  - Store data in a way that's widely useful to multiple stakeholders



1NF

Customer Firstname	Customer Lastname	Item 1	Item 2
Joe	Bloggs	Baked beans	Bread

2NF

Customer Firstname	Customer Lastname	Item
Joe	Bloggs	Baked beans
Joe	Bloggs	Bread

3NF

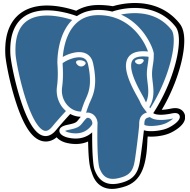
Customer ID	Customer Firstname	Customer Lastname
1	Joe	Bloggs
2	Jeff	Smith

Item ID	Item
1	Baked beans
2	Bread

Customer ID	Item
1	Baked beans
2	Bread

Common SQL Databases

- Several common SQL databases
  - PostgreSQL
  - SQL Server
  - Oracle
  - MySQL
- All store data in **relational databases**
  - Slight differences in SQL syntax used to retrieve and transform data
  - Slight differences in data types



SQLite

- SQLite is an open-source standalone database management system
    - Easy tool to transition from using flat files (CSVs)
    - Can be setup locally, with easy file connection and creation

SQLite:
https://www.sqlitetutorial.net/download-install-sqlite/

SQLite Studio:
https://github.com/pawelsalawa/sqlitestudio/releases

SQL Syntax

- SQL allows you to `SELECT` information `FROM` a source (i.e., a table)
  - Can be used as a calculator (first example)
  - Can select ***specific columns*** from a table
  - Can select ***all columns*** from a table (`SELECT *`)

```
SELECT 1;

SELECT
    id,
    country_short,
    country
FROM countries;

SELECT *
FROM countries;
```

SQL Syntax

SQL commands allow you to filter, sort, combine, and manipulate data

- Filter using the `WHERE` clause
  - Filter with mathematical operators
    `WHERE value >= 10`
  - Filter based on text values or patterns
    `WHERE name = 'USA'`
    `WHERE name like '%e%'`
  - Filter based on a list
    `WHERE value IN (7, 14, 21, 28)`
- ***Tip:** use `LIMIT 10;` to limit the number of records you return

```
SELECT
    id,
    country_short,
    country
FROM countries
WHERE country_short like 'E%'
LIMIT 10;
```

	id	country_short	country
1	44	ECU	Ecuador
2	45	EGY	Egypt
3	48	ERI	Eritrea
4	49	EST	Estonia
5	50	ETH	Ethiopia
6	136	ESP	Spain
7	161	ESH	Western Sahara

Activity 1

The `global_powerplants.db` database file contains data on the characteristics and output of thousands of power plants around the world.

1. Select the first few records from the `countries` and `power_plants` tables.
    - Do you see any information shared between the two tables? Note this down, as this will be necessary for the next section.
2. Identify the other tables available in this database by querying the `sqlite_master` schema.

Aggregating Information

- Aggregate information and calculate counts, averages, minimums, maximums, etc.
  - Count the number of records in a table
    ```
    SELECT count(*)
    FROM countries;
    ```
  - Find the number of records per group
- `GROUP BY` *all* columns not being aggregated
- ***Tip:** alias a column using `AS`

```
SELECT
    country_id,
    count(gppd_idnr) AS power_plants
FROM power_plants
GROUP BY country_id
ORDER BY country_id
LIMIT 10;
```

	country_id	power_plants
1	1	7
2	2	8
3	3	32
4	4	14
5	5	2
6	6	231
7	7	8
8	8	429

Filtering Results

- Filter results *before* or *after* an aggregation
  - `WHERE` filters the raw table contents before the `GROUP BY`
  - `HAVING` filters results based on the values in the aggregate clause
- ***Tip:*** Use `SELECT DISTINCT` or `count(DISTINCT column)` to get unique records

```
SELECT
    country_id,
    count(gppd_idnr) AS power_plants
FROM power_plants
WHERE wepp_id IS NOT NULL
GROUP BY country_id
HAVING count(gppd_idnr) > 500
LIMIT 10;
```

	country_id	power_plants
1	20	794
2	27	838
3	31	1480
4	53	613
5	68	652
6	155	1095
7	156	5218

Activity 2

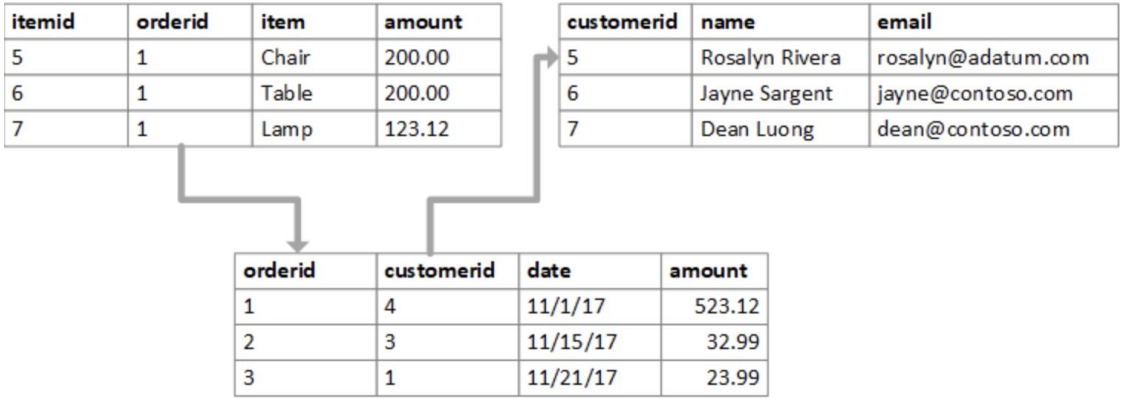
1. Determine how many records are in the `power_plants` table.
2. How many unique types of `primary_fuel` exist in the `power_plants` table?
3. On average, which type of `primary_fuel` has the greatest `capacity_mw`?
4. Which `country_id` has the highest total `generation_gwh` in the `power_generation` table?

Combining Data

The greatest value that SQL provides is the ability to easily join data from multiple tables.

Joins

- Combine information from multiple tables
  - Columns with shared information across tables are joined as *keys*
  - Type of join determines what's in the final table



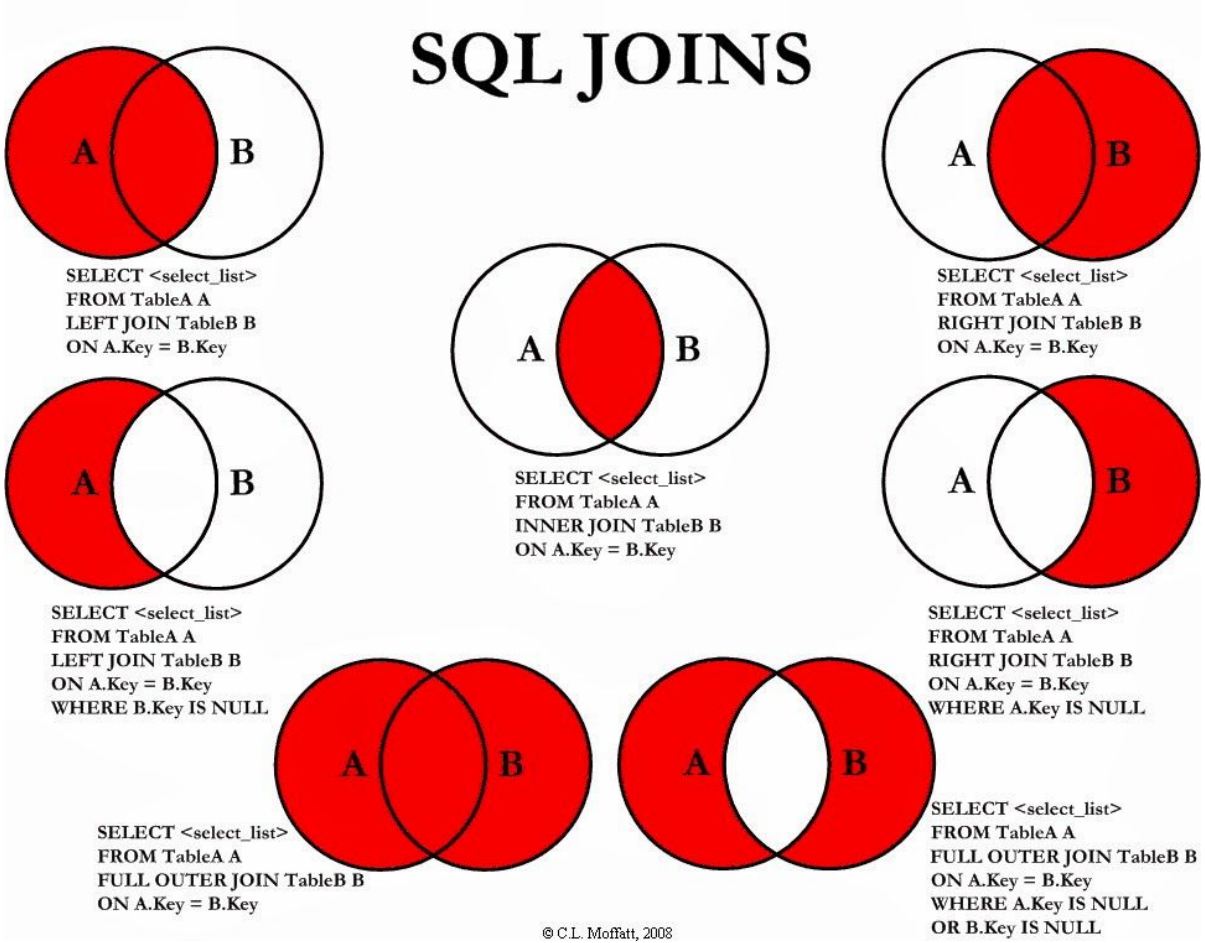
itemid	orderid	item	amount
5	1	Chair	200.00
6	1	Table	200.00
7	1	Lamp	123.12

customerid	name	email
5	Rosalyn Rivera	rosalyn@adatum.com
6	Jayne Sargent	jayne@contoso.com
7	Dean Luong	dean@contoso.com

orderid	customerid	date	amount
1	4	11/1/17	523.12
2	3	11/15/17	32.99
3	1	11/21/17	23.99

Joins

- Four main types
  - LEFT, RIGHT, OUTER, INNER
- ***Tip:*** Most commonly used joins are LEFT and INNER



Left Joins

- Retrieve all records in the left table and any in the right table with a value in the first
  - Specify your left table first
  - Give each table an alias
  - Prefix each column with the alias of its table

```
SELECT
    c.id,
    c.country,
    p.gppd_idnr,
    p.name,
    p.capacity_mw
FROM countries AS c
LEFT JOIN power_plants AS p
ON c.id = p.country_id
LIMIT 10;
```

	id	country	gppd_idnr	name	capacity_mw
1	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33
2	1	Afghanistan	GEODB0040541	Mahipar Hydroelectric Power Plan...	66
3	1	Afghanistan	GEODB0040534	Naghlu Dam Hydroelectric Power ...	100
4	1	Afghanistan	GEODB0040536	Nangarhar (Darunta) Hydroelectri...	11.55
5	1	Afghanistan	GEODB0040540	Northwest Kabul Power Plant Afg...	42
6	1	Afghanistan	GEODB0040537	Pul-e-Khumri Hydroelectric Powe...	6
7	1	Afghanistan	GEODB0040535	Sarobi Dam Hydroelectric Power ...	22
8	2	Albania	WRI1002169	Bistrica 1	27

Activity 3

1. What type of join do you need if you want the number of power plants per country, including countries with no records in the `power_plants` table?
2. Select all information from the `countries` and `power_plants` tables using a left join. Limit your results to 100.

Inner Joins

- Retrieve only records that share a key value in *both* tables
  - Specify either table first
  - Only records with a matching ID in both tables

```
SELECT
    c.id,
    c.country,
    p.gppd_idnr,
    p.name,
    p.capacity_mw
FROM countries AS c
INNER JOIN power_plants AS p
ON c.id = p.country_id
LIMIT 10;
```

	id	country	gppd_idnr	name	capacity_mw
1	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33
2	1	Afghanistan	GEODB0040541	Mahipar Hydroelectric Power Plan...	66
3	1	Afghanistan	GEODB0040534	Naghlu Dam Hydroelectric Power ...	100
4	1	Afghanistan	GEODB0040536	Nangarhar (Darunta) Hydroelectri...	11.55
5	1	Afghanistan	GEODB0040540	Northwest Kabul Power Plant Afg...	42
6	1	Afghanistan	GEODB0040537	Pul-e-Khumri Hydroelectric Powe...	6
7	1	Afghanistan	GEODB0040535	Sarobi Dam Hydroelectric Power ...	22
8	2	Albania	WRI1002169	Bistrica 1	27

Activity 4

1. How many power plants are in each country? Join the `power_plants` table to the `countries` table to get the country name.
2. Which country has the most power plants in the database? You can use `DESC` to sort your results in descending order.
3. Which country has the most unique sources in the `data_sources` table?

Combining Information In Columns

- Mathematical calculations on 1 or more columns without aggregations
  - Convert units
  - Calculate proportions/averages between columns
  - Manually calculate averages across multiple columns

```
SELECT
    country_id,
    year,
    generation_gwh * 1000 AS generation_mwh
FROM power_generation
LIMIT 10;
```

	country_id	year	generation_mwh
1	8	2013	89595.27777777778
2	8	2013	1095676.9444444445
3	8	2013	204804.4444444444
4	8	2013	7655.277777777778
5	8	2013	132456.6666666667
6	8	2013	4194.444444444444
7	8	2013	11468.333333333336
8	8	2013	180463.6111111111

Combining Information in Columns

- Manipulate text in columns
  - Concatenate multiple columns
  - Generate new columns from pieces of data
- ***Tip:*** Different versions of SQL (i.e., PostgreSQL vs. SQL Server) have different text manipulation functions, but most versions offer the same types of actions

```
SELECT
    id,
    source,
    geolocation_source,
    source || geolocation_source AS source_geolocation
FROM data_sources
LIMIT 10;
```

	id	source	geolocation_source	source_geolocation
1	1	4C Offshore	WRI	4C OffshoreWRI
2	1	4C Offshore	WRI	4C OffshoreWRI
3	1	4C Offshore	WRI	4C OffshoreWRI
4	1	4C Offshore	WRI	4C OffshoreWRI
5	1	4C Offshore	WRI	4C OffshoreWRI
6	6	9ren	Industry About	9renIndustry About
7	6	9ren	Industry About	9renIndustry About
8	6	9ren	Industry About	9renIndustry About

Activity 5

1. What is the oldest power plant in the `power_plants` table? Use the `commissioning_year` column to find out.
2. Create a new column called `fuels` that combines the `primary_fuel` and `other_fuel`.
3. Trim down the `url` column in the `data_sources` table using the `REPLACE()` function to remove `http://`. You can read about how it works here: https://www.sqlitetutorial.net/sqlite-replace-function/

Wrapping Up Joins

- You can join together as many tables as you want in multiple join combinations
    - Decide which keys you need to join to which table. Insufficient keys may lead to duplicates or errors!
    - The more joins you add, the slower your query will run.

```
SELECT
    c.id,
    c.country,
    p.gppd_idnr,
    p.name,
    p.capacity_mw,
    ds.source,
    ds.url
FROM countries AS c
INNER JOIN power_plants AS p
ON c.id = p.country_id
LEFT JOIN data_sources AS ds
ON p.source_id = ds.id
LIMIT 10;
```

	id	country	gppd_idnr	name	capacity_mw	source	url
1	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33	GEODB	http://globalenergyobservatory.org
2	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33	GEODB	http://globalenergyobservatory.org/form.p
3	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33	GEODB	http://globalenergyobservatory.org/form.p
4	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33	GEODB	http://globalenergyobservatory.org/form.p
5	1	Afghanistan	GEODB0040538	Kajaki Hydroelectric Power Plant ...	33	GEODB	http://globalenergyobservatory.org/form.p

Break time!

15 minutes

Transforming Your Data for Analysis

Generate the information you need to answer business questions in SQL

Conditional Statements

- SQL's `CASE` is similar to other languages' `IF/ELSE` statements
  - Can have multiple `WHEN` statements, which are evaluated *one at a time*
  - `ELSE NULL` is the default and can be left out
- ***Tip:** You can put an entire `CASE` statement inside an aggregate function.

```
SELECT
    CASE WHEN something THEN something_new
         ELSE something_else END as new_column
FROM table;
```

```
SELECT
    CASE WHEN something THEN something_new
         WHEN something_else THEN something_new_too
         ELSE null END as new_column
FROM table;
```

Conditional Statements

- Create new columns without altering the database
- Filter data without using a **`WHERE`** clause
- Pivot data into columns

```
SELECT
    country_id,
    AVG(CASE WHEN year = '2013' THEN generation_gwh END) as generation_2013,
    AVG(CASE WHEN year = '2014' THEN generation_gwh END) as generation_2014,
    AVG(CASE WHEN year = '2015' THEN generation_gwh END) as generation_2015
FROM power_generation
GROUP BY country_id
LIMIT 100;
```

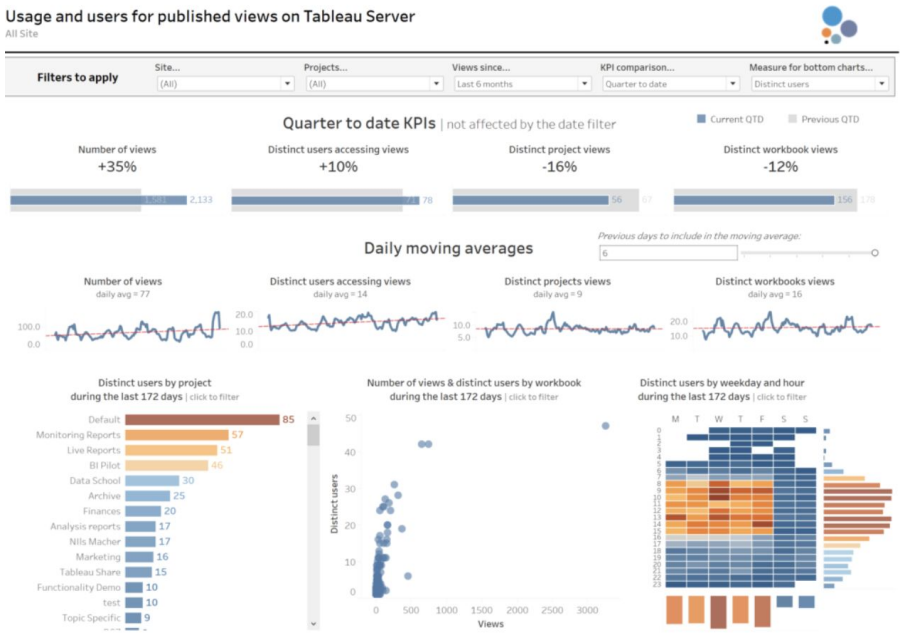
	country_id	generation_2013	generation_2014	generation_2015
1	8	869.6283003551495	812.0035930735933	885.661750645995
2	44	1145.0105555555556	*NULL*	*NULL*
3	68	2338.6823372747094	2455.203873159861	2338.6823372747094
4	102	197.35	*NULL*	*NULL*
5	156	568.9957661196626	529.0084653879596	571.2871734253833
6	159	3333.4444444444443	*NULL*	*NULL*

Activity 6

1. How many power plants in the database have a primary fuel that is a form of renewable energy -- `IN ('Hydro', 'Wind', 'Solar', 'Geothermal', 'Wave and Tidal')`?
2. There are a lot of missing values in the `year_of_capacity_data` column. Replace all missing values with the year `'2018'` using a `CASE` statement.

Answering Business Questions

- Stakeholder needs (dashboards, reports, etc.) and analytical projects guide a large proportion of queries and their underlying structure
    - Dictates which SQL skills you'll leverage most heavily
    - ***Example:*** At Greenhouse Software, we use Mode Analytics as our Business Intelligence tool and to track experiment results
        - A lot of `CASE` statements and subqueries



Answering Business Questions

- How well is an individual performing compared to a benchmark?
- How much has your key metric changed in the past 6 months?
- What's the 3 month rolling average of profit?

```
SELECT
    c.id,
    c.country,
    AVG(CASE WHEN other_fuel IS NOT NULL
             THEN 1 ELSE 0 END) AS pct_multiple_fuels
FROM countries c
LEFT JOIN power_plants p
ON c.id = p.country_id
GROUP BY
    c.id,
    c.country
LIMIT 10;
```

	id	country	pct_multiple_fuels
1	1	Afghanistan	0
2	2	Albania	0
3	3	Algeria	0.40625
4	4	Angola	0
5	5	Antarctica	0
6	6	Argentina	0.14718614718615
7	7	Armenia	0
8	8	Australia	0

SQL in the Real World

- Prepare SQL query results for further analysis in R, Python, etc.
- Run SQL queries directly in R or Python

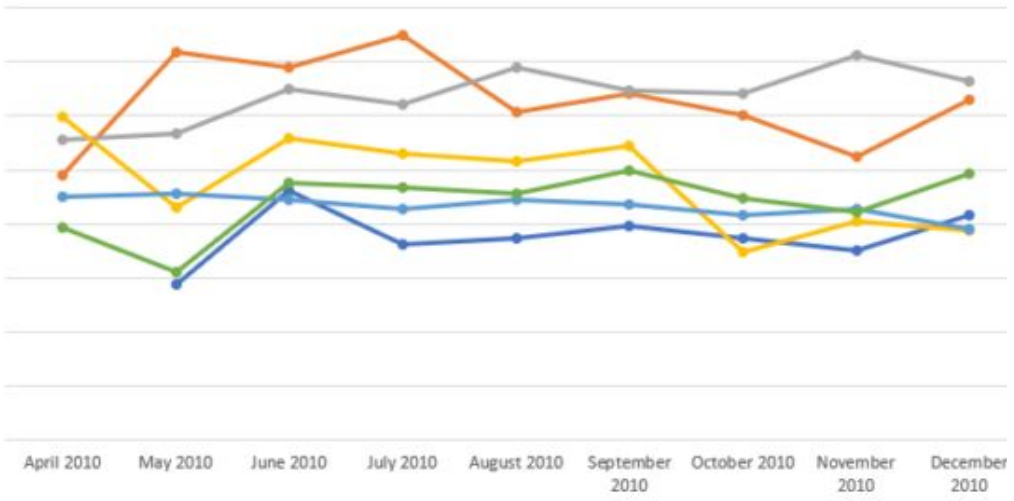
```
import pandas as pd
from sqlalchemy import create_engine

pplants = pd.read_csv('globalpowerplantdatabasev120/global_power_plant_database.csv')

# Create database
engine = create_engine('sqlite:///global_powerplants.db', echo=False)
pplants.to_sql('pplants_all', con = engine)

engine.execute("SELECT * FROM pplants_all LIMIT 5;").fetchall()

# Create and check unique country records
engine.execute("""
               CREATE TABLE countries_test AS
               SELECT DISTINCT country, country_long
               FROM pplants_all;
               """)

engine.execute("SELECT * FROM countries_test LIMIT 5;").fetchall()

# Create and check the countries table
engine.execute("""
               CREATE TABLE countries AS
               SELECT DISTINCT
                   RANK() over(ORDER BY country) as id,
                   country AS country_short,
                   country_long AS country
               FROM countries_test;
               """)

engine.execute("SELECT * FROM countries LIMIT 5;").fetchall()
```

Activity 7

1. Prepare a “pivot table” of countries’ average `capacity_mw` by whether or not the power plant’s primary fuel is a renewable `('Hydro', 'Wind', 'Solar', 'Geothermal', 'Wave and Tidal')`. Each renewable type should be a separate column.
2. Determine the average `capacity_mw` across all power plants. Then, calculate the average `capacity_mw` by `primary_fuel`. What % higher or lower is each fuel type capacity compared to your benchmark average?

Project Time!

- Prepare a final dataset to analyze the question: **Which types of power plants are generating more energy over time?**
  - Types of power plants = primary fuel
  - Time = power generation year
  - Energy = generation gigawatt hours in `power_generation`



Project Time!

- Prepare two queries: one examining the *descriptive statistics* (mean, min, max, sum), and one preparing a *long form* query (not pivoting into columns) for statistical tests
    - How many categories are in the `primary_fuel` column? Is this too many for a statistical test? Should we reduce it to renewable/non-renewable?
    - What do we do with null values?
- ***Note:*** SQLite does not have a function for calculating the standard deviation, while many other forms of SQL do.

Project Time!

```
SELECT
    CASE WHEN pp.primary_fuel IN ('Hydro', 'Wind', 'Solar', 'Geothermal', 'Wave and Tidal')
         THEN 'Renewable' ELSE 'Non-Renewable' END as power_type,
    pg.year,
    AVG(pg.generation_gwh) AS avg_gwh,
    SUM(pg.generation_gwh) AS total_gwh,
    MIN(pg.generation_gwh) AS min_gwh,
    MAX(pg.generation_gwh) AS max_gwh
FROM power_plants pp
INNER JOIN power_generation pg
ON pp.gppd_idnr = pg.gppd_idnr
WHERE generation_gwh IS NOT NULL
GROUP BY 1, 2
ORDER BY 1, 2;
```

	power_type	year	avg_gwh	total_gwh	min_gwh	max_gwh
1	Non-Renewable	2013	1067.5500348153432	4116472.934247963	-2.653	31431.08
2	Non-Renewable	2014	1092.886274786867	4307064.808935043	-262.902	32320.917
3	Non-Renewable	2015	1072.8516987240273	4072545.048356408	-2.653	31431.08
4	Non-Renewable	2016	1076.0736294624146	4069710.466626852	-2.653	31431.08
5	Non-Renewable	2017	1078.499655777763	4065943.702282167	-2.653	31431.08
6	Renewable	2013	225.5244017569	708146.621516666	-947.6	50834
7	Renewable	2014	182.7224308234097	639345.7854511106	-989.6189999999999	20261.569
8	Renewable	2015	195.51585403130687	607858.7901833331	-947.6	21073.181
9	Renewable	2016	195.5656286038693	607622.4080722219	-947.6	21073.181
10	Renewable	2017	195.80891724477365	607790.8791277774	-947.6	21073.181

Project Time!

```
SELECT
    CASE WHEN pp.primary_fuel IN ('Hydro', 'Wind', 'Solar', 'Geothermal', 'Wave and Tidal')
         THEN 'Renewable' ELSE 'Non-Renewable' END as power_type,
    pg.year,
    pg.generation_gwh
FROM power_plants pp
INNER JOIN power_generation pg
ON pp.gppd_idnr = pg.gppd_idnr
WHERE generation_gwh > 0
LIMIT 10;
```

	power_type	year	generation_gwh
1	Renewable	2013	89.59527777777778
2	Non-Renewable	2013	1095.6769444444444
3	Non-Renewable	2013	204.8044444444444
4	Non-Renewable	2013	7.65527777777778
5	Non-Renewable	2013	132.45666666666668
6	Non-Renewable	2013	4.19444444444444
7	Renewable	2013	11.46833333333334
8	Renewable	2013	180.4636111111111
9	Renewable	2013	323.6777777777778
10	Non-Renewable	2013	16593.38

Next Steps to Learn

- Topics
  - Query processing order
  - Subqueries
  - Common table expressions
- Datasets
  - European Soccer Database
  - NYC Jobs

Thank you!

- All materials (slides, activity sheet) will be uploaded to github.com/mona-kay/odsc-sql-for-data-science
- Follow me on twitter at mona_kay_