

# Module 2 - Session 3 - Data exploration

# Working effectively with data

CivicDataLab

2021/08/24 (updated: 2021-08-26)





Module 2 - Session - Data exploration

Working effectively with data

CivicDataLab

2021/08/11 (updated: 2021-08-

# Exercise - Exploring data from eCourts

civic data lab

**Dataset** - <u>Link</u> - *The database contains 81.2 million cases* 

Source: <u>Devdatalab</u>

#### **Objective**:

- Understand how the data is structured
- Import the data in a database
- Explore the sample datasets
- Find out the total cases present for each district for the year 2018

#### Tags

database <mark>large-datasets</mark> sqlite <mark>eCourts</mark>

## Exercise - Exploring data from eCourts

civic data lab

**Dataset** - <u>Link</u> - *The database contains 81.2 million cases* 

Source: <u>Devdatalab</u>

#### **Objective**:

- Understand how the data is structured
- Import the data in a database
- Explore the sample datasets
- Find out the total cases present for each district for the year 2018

#### Tags

#### database <mark>large-datasets</mark> sqlite eCourts

SELECT state\_code, dist\_code, count(\*) AS total\_cases FROM cases\_2018 GROUP BY state\_code, dist\_code

#### **Exercise - Using Databases**

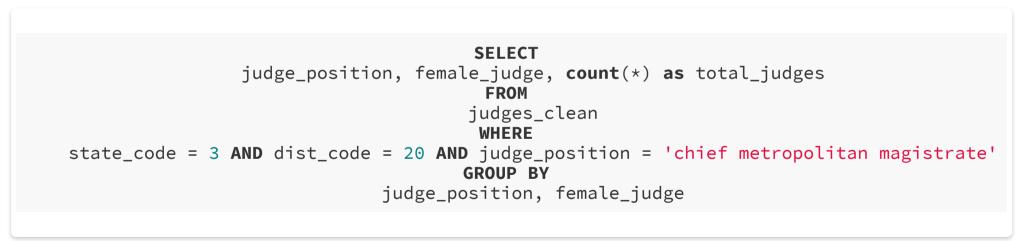
- Install SQLite DB Browser
- Create a new database
- Load the judges\_clean dataset in the DB
- Find the distribution of male/female judges in **Bengaluru** district court where judge position is *chief metropolitan magistrate*
- Save the file, as CSV, in the drive



### **Exercise - Using Databases**

Civic data lab

- Install SQLite DB Browser
- Create a new database
- Load the judges\_clean dataset in the DB
- Find the distribution of male/female judges in **Bengaluru** district court where judge position is *chief metropolitan magistrate*
- Save the file, as CSV, in the drive







Find the average duration of male and female judges appointed as chief metropolitan magistrate in the district courts of BENGALURU

# Working with Dates



Find the average duration of male and female judges appointed as chief metropolitan magistrate in the district courts of BENGALURU

```
SELECT
                    judge_position, female_judge, count(*) AS total_judges,
                                              avg(
 julianday(substr(end_date,7,4) || '-' || substr(end_date,4,2) || '-' || substr(end_date,1,2))
julianday(substr(start_date,7,4) || '-' || substr(start_date,4,2) || '-' || substr(start_date,1
                                      ) as avg_judge_duration
                                     FROM judges clean
                                           WHFRF
                                       state code = 3 AND
                                       dist code = 20 AND
                       judge position = 'chief metropolitan magistrate'
                                AND female_judge LIKE '%female%'
                                         GROUP BY
                                 judge_position, female_judge
                                         ORDER BY
                                    avg_judge_duration desc
```



# Working with SQL JOINS

# **JOINing Tables**



A JOIN command is used where we need to query data that is spread across multiple tables

Merging two data sets using SQL or SQL tools can be accomplished through JOINS. A JOIN is a SQL instruction in the FROM clause of your query that is used to identify the tables you are querying and how they should be combined.<sup>1</sup>

[1] Dataschool



# Table 1 Table 2 Outer Join 1 2 3 4 0uter Join

SELECT \* FROM facebook FULL OUTER JOIN linkedin ON facebook.name = linkedin.name

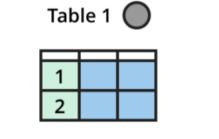
#### <u>Example</u>

4

INNER JOIN

**OUTER JOIN** 

NE	R JC	IN	



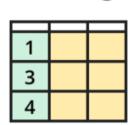


Table 2 🔘



1		

SELECT \* FROM facebook JOIN linkedin ON facebook.name = linkedin.name

<u>Example</u>



Table 1

**LEFT JOIN** 

**OUTER JOIN** 

SELECT \* FROM facebook LEFT JOIN linkedin ON facebook.name = linkedin.name

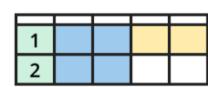
Table 2 🔘

1

3

4

<u>Example</u>



Left Join 🔘



INNER JOIN

LEFT JOIN

**UNION JOIN** 

**OUTER JOIN** 

SELECT FROM facebook UNION ALL SELECT FROM linkedin

Table 2 🔘

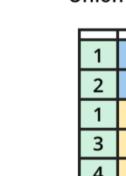
1

3

4

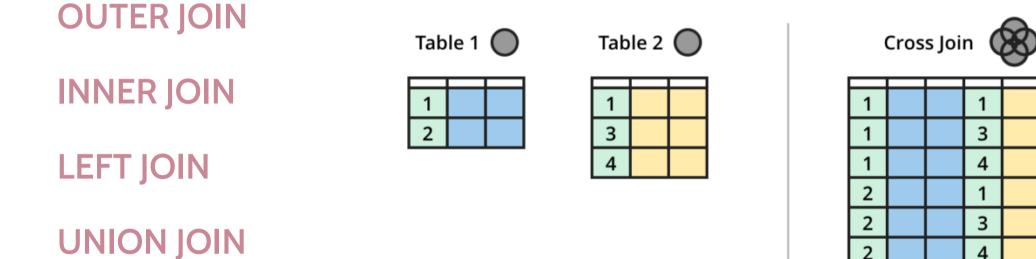
Table 1 🔘

2









**CROSS JOIN** 

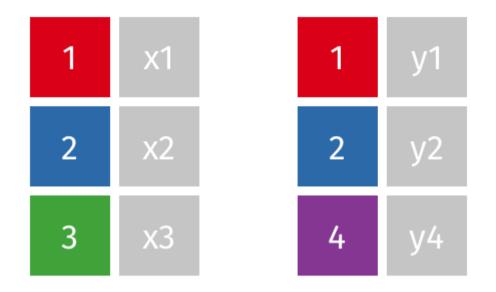
SELECT \* FROM facebook CROSS JOIN linkedin



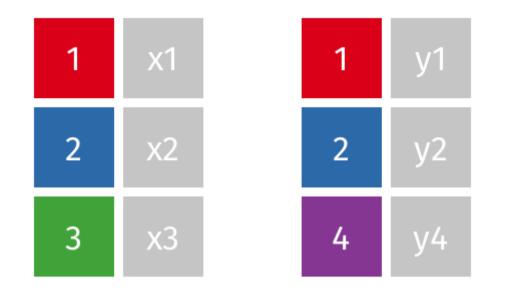




# JOIN - Quiz



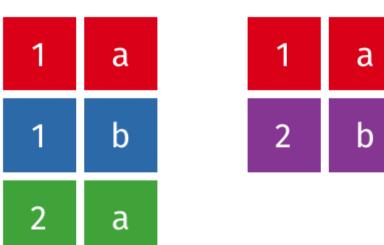




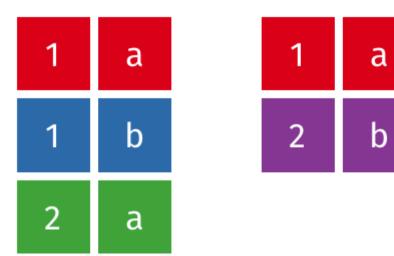






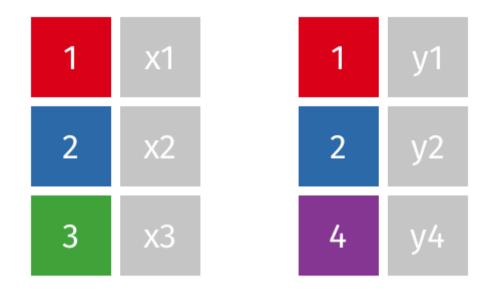






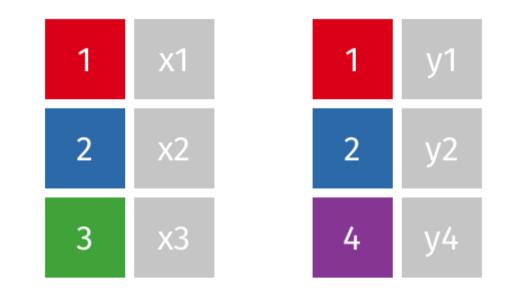
UNION

16 / 26











# **JOINS - Exercise 1**



- Create a table that only contains cases registered with the Karnataka district courts
- Join the above table with cases\_district\_key to get district name
- Find the total number of cases in each district. Arrange the results in descending order
- Use a subquery to combine the two queries in one

# **JOINS - Exercise 1**



- Create a table that only contains cases registered with the Karnataka district courts
- Join the above table with cases\_district\_key to get district name
- Find the total number of cases in each district. Arrange the results in descending order
- Use a subquery to combine the two queries in one



# **Other SQL Concepts**

#### **CASE WHEN**



**SELECT** City, **CASE WHEN** City = "SF" **THEN** "San Francisco" **ELSE** City **END** AS "Updated City" FROM friends

#### **CASE WHEN**



**SELECT** City, **CASE WHEN** City = "SF" **THEN** "San Francisco" **ELSE** City **END** AS "Updated City" FROM friends



select month, year, deaths, CASE WHEN deaths < 5000 THEN "lt 5K" WHEN 5000<=deaths<10000 THEN "5K-10K" WHEN deaths > 10000 THEN "gt10K" END as "trends" FROM mortality\_data; Example 2





Subquery in the **FROM clause** 

SELECT \* FROM (SELECT State, SUM (# of friends) FROM facebook GROUP BY state);





Subquery in the **FROM** clause

SELECT \* FROM (SELECT State, SUM (# of friends) FROM facebook GROUP BY state);

Subquery in the **WHERE clause** (*Returns single value*)

SELECT \* FROM facebook WHERE # of friends = (SELECT MAX(# of connections) FROM linkedin)





Subquery in the **FROM clause** 

SELECT \* FROM (SELECT State, SUM (# of friends) FROM facebook GROUP BY state);

Subquery in the **WHERE clause** (*Returns single value*)

SELECT \* FROM facebook WHERE # of friends = (SELECT MAX(# of connections) FROM linkedin)

Subquery in the WHERE clause (Returns multiple values)

SELECT \* FROM facebook WHERE # of friends IN (SELECT # of connections FROM linkedin)

## **EXERCISE - CASE WHEN & Subqueries**

civic data lab

- Load <u>Mortaliy data</u> in the database
- Create a column to tag months where the total number of deaths was above or below average for the state of Rajasthan.
- The column can have only two values *Above average* and *Below average*
- Sort the result dataset by year

# **EXERCISE - CASE WHEN & Subqueries**



- Load <u>Mortaliy data</u> in the database
- Create a column to tag months where the total number of deaths was above or below average for the state of Rajasthan.
- The column can have only two values *Above average* and *Below average*
- Sort the result dataset by year

select month, year, deaths, CASE WHEN deaths < (select avg(deaths) as avg\_deaths\_RJ from mortality\_data where state='Rajasthan') THEN "belowAvg" ELSE "aboveAvg" END as "trends" FROM mortality\_data where state='Rajasthan' order by year desc;

## JOINS - Exercise 2



Find the top 5 districts of Karnataka in terms of the number of cases that ended in conviction

## JOINS - Exercise 2

civic data lab

Find the top 5 districts of Karnataka in terms of the number of cases that ended in conviction

**SELECT** d.\*, e.district\_name **FROM** ( **SELECT** c.dist\_code, **count**(\*) **as** total\_convict\_cases FROM ( **SELECT** a.dist code, a.disp name, b.disp name s **FROM** cases 2018 karnataka AS a LEFT JOIN disp name key AS b ON a.disp name = b.disp name) AS c WHERE c.disp name s LIKE '%convict%' GROUP BY c.dist code) as d LEFT JOIN cases district key as e ON d.dist\_code = e.dist\_code WHERE e.state\_code = 3 **ORDER BY** total convict cases **DESC LIMIT** 5

# Regular Expressions (REGEX)



Regex, or Regular Expressions, is a sequence of characters, used to search and locate specific sequences of characters that match a pattern.

# Regular Expressions (REGEX)



Regex, or Regular Expressions, is a sequence of characters, used to search and locate specific sequences of characters that match a pattern.

#### The **LIKE** clause

Find all states that start with letter A

SELECT distinct state
FROM mortality\_data
WHERE state LIKE 'A%';

Find all states that end with word Pradesh

SELECT distinct state
FROM mortality\_data
WHERE state LIKE '%Pradesh';

#### **REGEX Exercise**

1. Import <u>NCRB data</u>

- 2. Find all crime heads related to children [can contain child or children]
- 3. Find all crime heads that mention Murder
- 4. Find all crime heads that start with Murder

5. Find all crime heads that are either SLL or IPC [ REGEXP / UNION ]





# **Queries and Feedback**