

# Module 2 - Session 3 - Data exploration

Working effectively with data

CivicDataLab

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

# SQL - Recap



## Module 2 - Session - Data exploration

Working effectively with  
data

CivicDataLab

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

# Exercise - Exploring data from eCourts

**Dataset** - [Link](#) - *The database contains 81.2 million cases*

**Source:** [Devdatalab](#)

## 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` `large-datasets` `sqlite` `eCourts`

# Exercise - Exploring data from eCourts

**Dataset** - [Link](#) - *The database contains 81.2 million cases*

**Source:** [Devdatalab](#)

## 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](#) [large-datasets](#) [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

- 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

```
SELECT
    judge_position, female_judge, count(*) as total_judges
FROM
    judges_clean
WHERE
    state_code = 3 AND dist_code = 20 AND judge_position = 'chief metropolitan magistrate'
GROUP BY
    judge_position, female_judge
```

# Working with Dates



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,1,2))
    ) as avg_judge_duration
FROM judges_clean
WHERE
    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](#)

# Type of JOINS

## OUTER JOIN

Table 1 ●

1		
2		

Table 2 ●

1		
3		
4		

Outer Join ●●

1				
2				
3				
4				


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

Example


# Type of JOINS

OUTER JOIN


INNER JOIN

Table 1 

1		
2		

Table 2 

1		
3		
4		

Inner Join 

1				

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


Example

# Type of JOINS


OUTER JOIN

INNER JOIN


LEFT JOIN

Table 1 

1		
2		

Table 2 

1		
3		
4		

Left Join 

1				
2				

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

Example

# Type of JOINS

OUTER JOIN

INNER JOIN

LEFT JOIN

UNION JOIN

Table 1 ●

1		
2		

Table 2 ●

1		
3		
4		

Union ●+●

1		
2		
1		
3		
4		

`SELECT FROM facebook UNION ALL SELECT FROM linkedin`

Example

# Type of JOINS


OUTER JOIN

INNER JOIN


LEFT JOIN

UNION JOIN

CROSS JOIN

Table 1 

1		
2		

Table 2 

1		
3		
4		

Cross Join 

1			1	
1			3	
1			4	
2			1	
2			3	
2			4	

```
SELECT * FROM facebook CROSS JOIN linkedin
```

Example





# JOIN - Quiz

1	x1
2	x2
3	x3

1	y1
2	y2
4	y4

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

**INNER JOIN**

1	a
1	b
2	a

1	a
2	b

1	a
1	b
2	a

1	a
2	b

**UNION**

1	x1
2	x2
3	x3

1	y1
2	y2
4	y4

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

**OUTER JOIN**

# 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

```
SELECT a2.district_name, count(*) AS total_cases
      FROM (SELECT
            a.*, b.district_name
            FROM
            cases_2018_karnataka AS a
            LEFT JOIN
            cases_district_key AS b
            ON a.state_code = b.state_code AND a.dist_code = b.dist_code) AS a2
      GROUP BY a2.district_name
      ORDER BY total_cases DESC
```

# 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
```

## Example 1

```
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

```
select month, year, deaths,
CASE
WHEN month = 1 THEN "Jan"
WHEN month = 2 THEN "Feb"
WHEN month = 3 THEN "Mar"
WHEN month = 4 THEN "Apr"
END as "monthName"
FROM mortality_data
WHERE
month <= 4 AND
year= 2019 AND
state="Rajasthan";
```

# Subqueries



## Subquery in the **FROM** clause

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

# Subqueries

## 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)
```

# Subqueries

## 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

- Load [Mortality data](#) 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 [Mortality data](#) 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

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 NCRB data
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