Module 2 - Session 4 - Data exploration

Working effectively with data

CivicDataLab

2021/09/01 (updated: 2021-09-02)

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.[1]

[1] Dataschool

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

civic data lab

- 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

CASE WHEN takes in values, checks them against a condition and **THEN** outputs values into a new column based on if it satisfies the condition.

CASE WHEN in SQL operates very similarly to “if then” statements in other programming languages.

CASE WHEN

CASE WHEN takes in values, checks them against a condition and **THEN** outputs values into a new column based on if it satisfies the condition.

CASE WHEN in SQL operates very similarly to “if then” statements in other programming languages.

Example

Create a new column as *defendant_type* as per the values given in the female_defendant column. Use these rules:

- Tag female defendants as *female*
- Tag male defendants as *male*
- Tag all other defendants as *not_sure*

CASE WHEN

CASE WHEN takes in values, checks them against a condition and **THEN** outputs values into a new column based on if it satisfies the condition.

CASE WHEN in SQL operates very similarly to “if then” statements in other programming languages.

Example

Create a new column as *defendant_type* as per the values given in the female_defendant column. Use these rules:

- Tag female defendants as *female*
- Tag male defendants as *male*
- Tag all other defendants as *not_sure*

Query

```
SELECT female_defendant,
CASE
    WHEN female_defendant = '1 female' THEN 'female
    WHEN female_defendant = '0 male' THEN 'male'
    ELSE 'not_sure'
END AS defendant_type
FROM cases_2018_karnataka
LIMIT 20
```

CASE WHEN - Examples

civic data lab

Example 1

Using the mortality dataset, categorise total number of deaths in a given month/year as *less than 5K* , *between 5K and 10K* and *greater than 10K*

```
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

On Mortality data, assign names for months where month <=4 in the year 2019

```
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

The core concept to grasp is that the subquery is a separate SQL query that produces a table that is then used in the main query.

Subqueries

The core concept to grasp is that the subquery is a separate SQL query that produces a table that is then used in the main query.

Objective

Find the total number of cases in BENGALURU where petitioner is a female aggregated by judge position (*Without Using JOINS*)

Subqueries

The core concept to grasp is that the subquery is a separate SQL query that produces a table that is then used in the main query.

Objective

Find the total number of cases in BENGALURU where petitioner is a female aggregated by judge position (*Without Using JOINS*)

Query

```
SELECT judge_position, count(*) AS total_cases
FROM cases_2018_karnataka
WHERE dist_code = (
    SELECT dist_code
    FROM cases_district_key
    WHERE district_name = 'BENGALURU'
    ) AND
    female_petitioner = '1 female'
GROUP BY judge_position
ORDER BY total_cases DESC
```

Subqueries - Examples

civic data lab

Subquery in the **FROM clause**

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

Subqueries - Examples

civic data lab

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 - Examples

civic data lab

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

civic data lab

- 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;
```

Window Functions

Window functions create a new column based on calculations performed on a subset or *window* of the data. This window starts at the first row on a particular column and increases in size unless you constrain the size of the window.

Window Functions

Window functions create a new column based on calculations performed on a subset or *window* of the data. This window starts at the first row on a particular column and increases in size unless you constrain the size of the window.

```
SELECT 'Day', 'Mile Driving',SUM('Miles Driving')
OVER(ORDER BY 'Day') AS 'Running Total'
FROM 'Running total mileage visual'
```

Window Functions

Window functions create a new column based on calculations performed on a subset or *window* of the data. This window starts at the first row on a particular column and increases in size unless you constrain the size of the window.

```
SELECT 'Day', 'Mile Driving',SUM('Miles Driving')
OVER(ORDER BY 'Day') AS 'Running Total'
FROM 'Running total mileage visual'
```

Running total mileage visual

Day	Miles Driving	Running Total
Jan. 1	60	
Jan. 2	80	
Jan. 3	10	
Jan. 4	150	

Window Functions - Use Cases

civic data lab

Creating additional columns

Using Mortality data, find if the total deaths in a state in a given month and year was above or below the average number of deaths in that year for a state

```
SELECT *,
    CASE
        WHEN deaths < avg_deaths THEN 'Below Average'
        ELSE 'Above Average'
    END AS trends
FROM (
    SELECT *, AVG(deaths) OVER(PARTITION BY state,year) as avg_deaths
    FROM mortality_data
    )
```

Window Functions - Use Cases

civic data lab

Ranking Items

Assign ranks as per the total cases registered under each judge position **across all districts**

```
SELECT *,
    RANK() over(ORDER BY total_cases DESC) AS ranking
FROM
    (
        SELECT dist_code, judge_position, count(*) AS total_cases
        FROM cases_2018_karnataka
        GROUP BY dist_code,judge_position
    )
```

PARTITION BY AND ORDER BY

civic data lab

Ranking with PARTITION BY

Ranking **within each district**

```
SELECT *,
    RANK() over(PARTITION BY dist_code ORDER BY total_cases DESC) AS ranking
FROM
    (
        SELECT dist_code, judge_position, count(*) AS total_cases
        FROM cases_2018_karnataka
        GROUP BY dist_code,judge_position
    )
```

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)

civic data lab

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*]

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
```

Queries and Feedback