Spring Board 8.2.3 DataCamp Intermediate SQL Exercises

cihat kurt[1]

[1]Texas A&M University

June 6, 2022

Basic CASE statements

What is your favorite team?

The *European Soccer Database* contains data about 12,800 matches from 11 countries played between 2011-2015! Throughout this course, you will be shown filtered versions of the tables in this database in order to better explore their contents.

In this exercise, you will identify matches played between *FC Schalke 04* and *FC Bayern Munich*. There are 2 teams identified in each match in the `hometeam_id` and `awayteam_id` columns, available to you in the filtered `matches_germany` table. ID can join to the `team_api_id` column in the `teams_germany` table, but you cannot perform a join on both at the same time.

However, you can perform this operation using a `CASE` statement once you've identified the `team_api_id` associated with each team!

- Create a `CASE` statement that identifies whether a match in Germany included `FC Bayern Munich`, `FC Schalke 04`, or neither as the home team.
- Group the query by the `CASE` statement alias, `home_team`.

```
-- Identify the home team as Bayern Munich, Schalke 04, or neither
SELECT
    CASE WHEN hometeam_id = 10189 THEN 'FC Schalke 04'
        WHEN hometeam_id = 9823 THEN 'FC Bayern Munich'
         ELSE 'Other' END AS home_team,
    COUNT(id) AS total_matches
FROM matches_germany
-- Group by the CASE statement alias
GROUP BY home_team;
```

home_team	total_matches
FC Bayern Munich	68
Other	1088
FC Schalke 04	68

CASE statements comparing column values

Barcelona is considered one of the strongest teams in Spain's soccer league.

In this exercise, you will be creating a list of matches in the 2011/2012 season where Barcelona was the *home team*. You will do this using a `CASE` statement that compares the values of two columns to create a new group – wins, losses, and ties.

In 3 steps, you will build a query that identifies a match's winner, identifies the identity of the opponent, and finally filters for Barcelona as the home team. Completing a query in this order will allow you to watch your results take shape with each new piece of information.

The `matches_spain` table currently contains Barcelona's matches from the 2011/2012 season, and has two key columns, `hometeam_id` and `awayteam_id`, that can be joined with the `teams_spain` table. However, you can only join `teams_spain` to one column at a time.

- Left join the `teams_spain` table `team_api_id` column to the `matches_spain` table `awayteam_id`. This allows us to retrieve the *away* team's identity.
- Select `team_long_name` from `teams_spain` as `opponent` and complete the `CASE` statement from Step 1.

```
SELECT
    m.date,
    --Select the team long name column and call it 'opponent'
    t.team_long_name AS opponent,
    -- Complete the CASE statement with an alias
    CASE WHEN m.home_goal > away_goal THEN 'Home win!'
        WHEN m.home_goal < away_goal THEN 'Home loss :('
        ELSE 'Tie' END AS outcome
FROM matches_spain AS m
-- Left join teams_spain onto matches_spain
LEFT JOIN teams_spain AS t
ON m.awayteam_id = t.team_api_id;
-- Filter for Barcelona as the home team
WHERE m.hometeam_id = 8634;
```

date	opponent	outcome
2011-10-29	RCD Mallorca	Barcelona win!
2011-11-19	Real Zaragoza	Barcelona win!
2011-12-03	Levante UD	Barcelona win!

CASE Advance

EX: Multiple Cases

Barcelona and Real Madrid have been rival teams for more than 80 years. Matches between these two teams are given the name *El Clásico* (The Classic). In this exercise, you will query a list of matches played between these two rivals.

You will notice in Step 2 that when you have multiple logical conditions in a `CASE` statement, you may quickly end up with a large number of `WHEN` clauses to logically test every outcome you are interested in. It's important to make sure you don't accidentally exclude key information in your `ELSE` clause.

In this exercise, you will retrieve information about matches played between **Barcelona** (id = 8634) and **Real Madrid** (id = 8633). Note that the query you are provided with already identifies the *Clásico* matches using a filter in the `WHERE` clause.

```
SELECT
    date,
    CASE WHEN hometeam_id = 8634 THEN 'FC Barcelona'
         ELSE 'Real Madrid CF' END as home,
    CASE WHEN awayteam_id = 8634 THEN 'FC Barcelona'
         ELSE 'Real Madrid CF' END as away,
    -- Identify all possible match outcomes
    CASE WHEN home_goal > away_goal AND hometeam_id = 8634 THEN 'Barcelona win!'
        WHEN home_goal > away_goal AND hometeam_id = 8633 THEN 'Real Madrid win!'
        WHEN home_goal < away_goal AND awayteam_id = 8634 THEN 'Barcelona win!'
        WHEN home_goal < away_goal AND awayteam_id = 8633 THEN 'Real Madrid win!'
        ELSE 'Tie!' END AS outcome
FROM matches_spain
WHERE (awayteam_id = 8634 OR hometeam_id = 8634)
      AND (awayteam_id = 8633 OR hometeam_id = 8633);
```

date	home	away	outcome
2011-12-10	Real Madrid CF	FC Barcelona	Barcelona win!
2012-04-21	FC Barcelona	Real Madrid CF	Real Madrid win!
2013-03-02	Real Madrid CF	FC Barcelona	Real Madrid win!
2012-10-07	FC Barcelona	Real Madrid CF	Tie!

Filtering your CASE statement

`CASE` statements allow you to categorize data that you're interested in – and exclude data you're not interested in. In order to do this, you can use a `CASE` statement as a filter in the `WHERE` statement to remove output you don't want to see.

Here is how you might set that up:

In essence, you can use the `CASE` statement as a filtering column like any other column in your database. The only difference is that you *don't* alias the statement in `WHERE`.

```
SELECT *
FROM table
WHERE
    CASE WHEN a > 5 THEN 'Keep'
         WHEN a <= 5 THEN 'Exclude' END = 'Keep';
```

Ex: The first code shows the usage of case in **SELECT** statement. And second code uses case in WHERE caluse to filter only BOLOGNO wins and also get rid of null values

```
-- Select the season and date columns
SELECT
```

```
    season,
    date,
    -- Identify when Bologna won a match
    CASE WHEN hometeam_id = 9857 AND home_goal > away_goal THEN 'Bologna Win'
        WHEN awayteam_id = 9857 AND away_goal > home_goal THEN 'Bologna Win'
        END AS outcome
FROM matches_italy;
```

```
-- Select the season, date, home_goal, and away_goal columns
SELECT
    season,
    date,
    home_goal,
    away_goal
FROM matches_italy
WHERE
-- Exclude games not won by Bologna
    CASE WHEN hometeam_id = 9857 AND home_goal > away_goal THEN 'Bologna Win'
         WHEN awayteam_id = 9857 AND away_goal > home_goal THEN 'Bologna Win'
         END IS NOT NULL;
```

Notice in the second query CASE statement used only to filter data, but it does not create any column or values. Thus, labels used in it ('Bologno Win') are useless for this particular case because we don't want to filter based on labels but necessary for the syntax.

COUNT using CASE WHEN

Do the number of soccer matches played in a given European country differ across seasons? We will use the European Soccer Database to answer this question.

You will examine the number of matches played in 3 seasons within each country listed in the database. This is much easier to explore with each season's matches in separate columns. Using the `country` and unfiltered `match` table, you will count the number of matches played in each country during the 2012/2013, 2013/2014, and 2014/2015 match seasons.

```
SELECT
    c.name AS country,
    -- Count matches in each of the 3 seasons
    COUNT(CASE WHEN m.season = '2012/2013' AND
       m.home_goal > m.away_goal THEN m.id END) AS matches_2012_2013,
    COUNT(CASE WHEN m.season = '2013/2014' AND
       m.home_goal > m.away_goal THEN m.id END) AS matches_2013_2014,
    COUNT(CASE WHEN m.season = '2014/2015' AND
       m.home_goal > m.away_goal THEN m.id END) AS matches_2014_2015
FROM country AS c
LEFT JOIN match AS m
ON c.id = m.country_id
-- Group by country name alias
GROUP BY country;
```

country	matches_2012_2013	matches_2013_2014	matches_2014_2015
Portugal	103	108	137
France	170	168	181
Scotland	89	102	102
Netherlands	137	144	138
Spain	189	179	171
Belgium	102	6	106
Italy	177	181	152
Germany	130	145	145
England	166	179	172
Switzerland	84	82	76
Poland	97	110	114

Same output can be obtained using SUM

```
SELECT
    c.name AS country,
    -- Sum the total records in each season where the home team won
    SUM(CASE WHEN m.season = '2012/2013' AND m.home_goal > m.away_goal
        THEN 1 ELSE 0 END) AS matches_2012_2013,
    SUM(CASE WHEN m.season = '2013/2014' AND m.home_goal > m.away_goal
        THEN 1 ELSE 0 END) AS matches_2013_2014,
    SUM(CASE WHEN m.season = '2014/2015' AND m.home_goal > m.away_goal
        THEN 1 ELSE 0 END) AS matches_2014_2015
FROM country AS c
LEFT JOIN match AS m
ON c.id = m.country_id
-- Group by country name alias
GROUP BY country;
```

Calculating percent with CASE and AVG

`CASE` statements will return any value you specify in your `THEN` clause. This is an incredibly powerful tool for robust calculations and data manipulation when used in conjunction with an aggregate statement. One key task you can perform is using `CASE` inside an `AVG` function to calculate a percentage of information in your database.

Here's an example of how you set that up:

```
AVG(CASE WHEN condition_is_met THEN 1
         WHEN condition_is_not_met THEN 0 END)
```

```
SELECT
    c.name AS country,
    -- Calculate the percentage of tied games in each season
    ROUND(AVG(CASE WHEN m.season='2013/2014' AND m.home_goal = m.away_goal THEN 1
            WHEN m.season='2013/2014' AND m.home_goal <> m.away_goal THEN 0
            END),2) AS ties_2013_2014,
    ROUND(AVG(CASE WHEN m.season='2014/2015' AND m.home_goal = m.away_goal THEN 1
            WHEN m.season='2014/2015' AND m.home_goal <> m.away_goal THEN 0
```

```
            END),2) AS ties_2014_2015
FROM country AS c
LEFT JOIN matches AS m
ON c.id = m.country_id
GROUP BY country;
```