



Advance SQL

Connect to mysql

```
/usr/local/mysql/bin/mysql -u root
```

Commonly used SQL Functions

 functions	 Property
<u>IFNULL</u>	Return a specified value if the expression is NULL, otherwise return the expression <i>Syntax :- IFNULL(expression, alt_value)</i>
<u>ISNULL</u>	Returns 1 or 0 depending on whether an expression is NULL <i>Syntax :- ISNULL(expression)</i>
<u>NULLIF</u>	Compares two expressions and returns NULL if they are equal. Otherwise, the first expression is returned <i>Syntax :- NULLIF(expr1, expr2)</i>

Script

```
create table person (id int, name varchar(10), email varchar(50));

insert into person values (100, null, "someting@gmail.com");

insert into person values (101, "Arjun", "someting@yahoo.com");
```

ISNULL:

```
SELECT ISNULL(350); //0

SELECT ISNULL(NULL); //1
select isnull(name) from person;
```

IFNULL:

```
select ifnull(name,"name not available") from person;
```

NULLIF:

```
mysql> select nullif(4,4); //NULL

select nullif(3,4); //3

select nullif(30,4); //30
```

The SQL CASE statement

The CASE statement is SQL's way of handling if/then logic.

The CASE statement is followed by at least one pair of WHEN and THEN statements

Every CASE statement must end with the END statement. The ELSE statement is optional, and provides a way to capture values not specified in the WHEN/THEN statements. CASE is easiest to understand in the context of an example:

Syntax 1:

```
SELECT  
CASE  
WHEN-THEN  
END AS
```

Syntax 2:

```
SELECT  
CASE  
WHEN-THEN  
ELSE /*ELSE is optional*/  
END AS
```

Write a query to tell if the players are senior or not.

```
select player_name, year,  
CASE  
WHEN year='SR' THEN 'yes'  
else NULL  
END  
AS IS_A_SR FROM college_football_players;
```

plain English, here's what's happening:

1. The CASE statement checks each row to see if the conditional statement—year = 'SR' is true.
2. For any given row, if that conditional statement is true, the word "yes" gets printed in the column that we have named is_a_senior.
3. In any row for which the conditional statement is false, nothing happens in that row, leaving a null value in the is_a_senior column.
4. At the same time all this is happening, SQL is retrieving and displaying all the values in the player_name and year columns.

The above query makes it pretty easy to see what's happening because we've included the CASE statement along with the year column itself. You can check each row to see whether year meets the condition year = 'SR' and then see the result in the column generated using the CASE statement.

But what if you don't want null values in the is_a_senior column? The following query replaces those nulls with "no":

```
select player_name, year,  
CASE WHEN year='SR' THEN 'yes'  
else 'no'  
END  
AS IS_A_SR FROM college_football_players;
```

Adding multiple conditions to a CASE statement

You can also define a number of outcomes in a CASE statement by including as many WHEN/THEN statements as you'd like:

```
SELECT player_name, weight,
       CASE WHEN weight > 250 THEN 'over 250'
            WHEN weight > 200 THEN '201-250'
            WHEN weight > 175 THEN '176-200'
            ELSE '175 or under' END AS weight_group
FROM college.college_football_players;
```

In the above example, the WHEN/THEN statements will get evaluated in the order that they're written. So if the value in the weight column of a given row is 300, it will produce a result of "over 250." Here's what happens if the value in the weight column is 180, SQL will do the following:

1. Check to see if weight is greater than 250. 180 is not greater than 250, so move on to the next WHEN/THEN
2. Check to see if weight is greater than 200. 180 is not greater than 200, so move on to the next WHEN/THEN
3. Check to see if weight is greater than 175. 180 is greater than 175, so record "175-200" in the weight_group column.

You can also string together multiple conditional statements with AND and OR the same way you might in a WHERE clause:

```
SELECT player_name,
       CASE WHEN year = 'FR' AND position = 'WR' THEN 'frosh_wr'
            ELSE NULL END AS sample_case_statement
FROM college.college_football_players;
```

POINTS TO KEEP IN NOTICE:

1. The CASE statement always goes in the SELECT clause
2. CASE must include the following components: WHEN, THEN, and END. ELSE is an optional component.
3. You can make any conditional statement using any conditional operator (like WHERE) between WHEN and THEN. This includes stringing together multiple conditional statements using AND and OR.

You can include multiple WHEN statements, as well as an ELSE statement to deal with any unaddressed conditions.

Using CASE inside of aggregate functions

In the previous examples, data was displayed vertically, but in some instances, you might want to show data horizontally. This is known as "pivoting".

Let's take the following query

```
SELECT CASE WHEN year = 'FR' THEN 'FR'
           WHEN year = 'SO' THEN 'SO'
           WHEN year = 'JR' THEN 'JR'
           WHEN year = 'SR' THEN 'SR'
           ELSE 'No Year Data' END AS year_group,
       COUNT(1) AS count
FROM college.college_football_players
GROUP BY 1
```

Group by n groups by column number n.

And re-orient it horizontally:

```
SELECT COUNT(CASE WHEN year = 'FR' THEN 1 ELSE NULL END) AS fr_count,
       COUNT(CASE WHEN year = 'SO' THEN 1 ELSE NULL END) AS so_count,
       COUNT(CASE WHEN year = 'JR' THEN 1 ELSE NULL END) AS jr_count,
       COUNT(CASE WHEN year = 'SR' THEN 1 ELSE NULL END) AS sr_count
FROM college.college_football_players
```

Script

This script will create a database "college" and inside that we will have 2 tables college_football_players and dc_bikeshare_q1_2012 ;

```
Create database college;
use college;

create table college_football_players (full_school_name varchar(250), school_name varchar(250), player_name varchar(250), position varchar(250), height int, weight int, year varchar(250), hometown varchar(250), state varchar(250), id int);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Ralph Abernathy", "RB", 67, 161, "SR", "ATLANTA, GA", "GA", 1);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Mekale McKay", "WR", 78, 195, "SO", "LOUISVILLE, KY", "KY", 2);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Trenier", "CB", 71, 255, "SO", "WINTER GARDEN, FL", "FL", 3);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Bennie Coney", "WR", 75, 216, "FR", "PLANT CITY, FL", "FL", 4);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Johnny Holton", "WR", 75, 190, "SR", "MIAMI, FL", "FL", 5);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Howard Wilder", "WR", 71, 180, "FR", "SEA ISLAND, GA", "GA", 6);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Howard Wilder1", "WR", 71, 180, "JR", "SEA ISLAND, GA", "GA", 7);

insert into college_football_players (full_school_name, school_name, player_name, position, height, weight, year, hometown, state, id)
values("Cincinnati Bearcats", "Cincinnati", "Howard Wilder2", "WR", 71, 180, "JR", "SEA ISLAND, GA", "GA", 8);
-----

create table dc_bikeshare_q1_2012( duration varchar(250), duration_seconds int, start_time DATETIME, start_station varchar(250), start_terminal varchar(250), end_time DATETIME, end_station varchar(250), end_terminal varchar(250));

insert into dc_bikeshare_q1_2012 (duration, duration_seconds, start_time, start_station, start_terminal, end_time, end_station, end_terminal)
values("0h 7m 55sec.", 475, "2012-01-01 00:04:02", "7th & R St NW / Shaw Library", 31245, "2012-01-01 00:11:00", "7th & T St NW", 31109, "W0");

insert into dc_bikeshare_q1_2012 (duration, duration_seconds, start_time, start_station, start_terminal, end_time, end_station, end_terminal)
values("0h 19m 22sec", 1162, "2012-01-01 00:05:00", "Georgia & New Hampshire Ave NW", 31400, "2012-01-01 00:29:00", "16th & Harvard St", 31602, "W0");

insert into dc_bikeshare_q1_2012 (duration, duration_seconds, start_time, start_station, start_terminal, end_time, end_station, end_terminal)
values("0h 19m 5sec.", 1145, "2012-01-01 00:06:00", "Georgia & New Hampshire Ave NW", 31400, "2012-01-01 00:29:00", "16th & Harvard St", 31602, "W0");

insert into dc_bikeshare_q1_2012 (duration, duration_seconds, start_time, start_station, start_terminal, end_time, end_station, end_terminal)
values("0h 8m 5sec.", 485, "2012-01-01 00:10:00", "14th & V St NW", 31102, "2012-01-01 00:23:00", "Park Rd & Holmead Pl NW", 31602, "W0");

insert into dc_bikeshare_q1_2012 (duration, duration_seconds, start_time, start_station, start_terminal, end_time, end_station, end_terminal)
values("0h 7m 51sec.", 471, "2012-01-01 00:15:00", "11th & Kenyon St NW", 31102, "2012-01-01 00:23:00", "7th & T St NW", 31109, "W0");

insert into dc_bikeshare_q1_2012 (duration, duration_seconds, start_time, start_station, start_terminal, end_time, end_station, end_terminal)
values("0h 5m 58sec.", 358, "2012-01-01 00:17:00", "Court House Metro / Wilson Blvd & N Uhle St", 31102, "2012-01-01 00:23:00", "Lynn & N", 31109, "W0");
```

End of Script