MySQL | Regular expressions (Regexp)

shreyanshi_arun

MySQL supports another type of pattern matching operation based on the regular expressions and the REGEXP operator.

1. It provide a powerful and flexible pattern match that can help us implement power search utilities for our database systems.
2. REGEXP is the operator used when performing regular expression pattern matches. RLIKE is the synonym.
3. It also supports a number of metacharacters which allow more flexibility and control when performing pattern matching.
4. The backslash is used as an escape character. It's only considered in the pattern match if double backslashes have used.
5. Not case sensitive.

Pattern	What the Pattern matches
*	Zero or more instances of string preceding it
+	One or more instances of strings preceding it

?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets
[A-Z]	match any upper case letter.

Pattern	What the Pattern matches
[a-z]	match any lower case letter
[0-9]	match any digit from 0 through to 9.
[[:<:]]	matches the beginning of words.
[[:>:]]	matches the end of words.
[:class:]	matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters.
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element

Examples with explanation :

- **Match beginning of string(^):** Gives all the names starting with 'sa'.Example- sam,samarth.

```
SELECT name FROM student_tbl WHERE name REGEXP '^sa';
```

- **Match the end of a string($):** Gives all the names ending with 'on'.Example – norton,merton.

```
SELECT name FROM student_tbl WHERE name REGEXP 'on$';
```

- **Match zero or one instance of the strings preceding it(?):** Gives all the titles containing 'com'.Example – comedy , romantic comedy.

```
SELECT title FROM movies_tbl WHERE title REGEXP 'com?';
```

- **matches any of the patterns p1, p2, or p3(p1|p2|p3):** Gives all the names containing 'be' or 'ae'.Example – Abel, Baer.

```
SELECT name FROM student_tbl WHERE name REGEXP 'be|ae' ;
```

- **Matches any character listed between the square brackets([abc]):** Gives all the names containing 'j' or 'z'.Example – Lorentz, Rajs.

```
SELECT name FROM student_tbl WHERE name REGEXP '[jz]' ;
```

- **Matches any lower case letter between ‘a’ to ‘z’- ([a-z]) ([a-z] and (.)):** Retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by the letter ‘a’.Example – Tobias, sewall. Matches any single character(.)

```
SELECT name FROM student_tbl WHERE name REGEXP '[b-g].[a]' ;
```

- **Matches any character not listed between the square brackets.([^abc]):** Gives all the names not containing ‘j’ or ‘z’. Example – nerton, sewall.

```
SELECT name FROM student_tbl WHERE name REGEXP '[^jz]' ;
```

- **Matches the end of words[[:>:]]:** Gives all the titles ending with character “ack”. Example – Black.

```
SELECT title FROM movies_tbl WHERE REGEXP 'ack[[:>:]]';
```

- **Matches the beginning of words[[:<:]]:** Gives all the titles starting with character “for”. Example – Forgetting Sarah Marshal.

```
SELECT title FROM movies_tbl WHERE title REGEXP '[[:<:]]for';
```

- **Matches a character class[:class:]:** i.e [:lower:]- lowercase character ,[:digit:] – digit characters etc. Gives all the titles containing alphabetic character only. Example – stranger things, Avengers.

```
SELECT title FROM movies_tbl WHERE REGEXP '[:alpha:]' ;
```

- **Matches the beginning of all words by any character listed between the square brackets. (^[abc]):** Gives all the names starting with ‘n’ or ‘s’. Example – nerton, sewall.

```
SELECT name FROM student_tbl WHERE name REGEXP '^[ns]' ;
```

Last Updated : 05 Sep, 2022

116

Similar Reads

1. MySQL | Recursive CTE (Common Table Expressions)
2. MySQL | Common MySQL Queries

IFNULL in MySQL

DevanshuAgarwal

Given a TABLE, in this TABLE, it prints entry of the table. If table is empty then it gives NULL.

Examples:

QUESTION : Given an employee table, print name from the given table which id equals to 2.

id	Name
1	Geek1
2	Geek2
3	Geek3
4	Geek4

Output : Geek2

QUESTION : Given same Employee table, print name from the given table which id equals to 5.
Output : NULL

AD

Approach: In this case, we use here IFNULL. IFNULL print the null if the table is an empty or other condition.

Query:-

```
SELECT
IFNULL(
  (SELECT NAME
  from employee
  where id = 2),
  'NULL') as NAME;


Output:-
Geek2
```

Query:-

```
SELECT
IFNULL(
  (SELECT NAME
  from employee
  where id = 5),
  'NULL') as NAME;


Output:-
NULL
```

Last Updated : 03 Mar, 2018

7

Similar Reads

1. MySQL | Common MySQL Queries
2. MySQL | LEAD() and LAG() Function
3. PHP | MySQL UPDATE Query
4. PHP | MySQL Database Introduction
5. PHP | MySQL (Creating Database)
6. PHP | MySQL (Creating Table)
7. PHP | Inserting into MySQL database

MySQL | LAST_DAY() Function

Shubrodeep Banerjee

The LAST_DAY() function in MySQL can be used to know the last day of the month for a given date or a datetime. The LAST_DAY() function takes a *date* value as argument and returns the last day of month in that *date*. The *date* argument represents a valid date or datetime.

Syntax:

```
LAST_DAY( Date );
```

If the date or datetime value is invalid, the function returns NULL.

Parameters Used:
Date: The LAST_DAY() function takes a single argument Date. It is the date or datetime value for which you want to extract the last day of the month.

AD

The LAST_DAY() function returns the last day of the month for a valid *Date* argument. If the argument *Date* is invalid or null, then the function will also return NULL.

Below are some common examples or usages of the LAST_DAY() function:

1. **Extracting last day from a given date**:
   To know the last date of the month December 2017,the LAST_DAY() function can be executed in the following way:
   Syntax :

```
mysql> SELECT LAST_DAY('2017-12-25');
```

Output :

```
'2017-12-31'
```

2. **Extracting the last day from a given datetime**:
   To know the last date of the month December using datetime format,the LAST_DAY() function can be executed in the following way:
   Syntax :

```
mysql> SELECT LAST_DAY('2017-12-25 08:21:05');
```

Output :

```
'2017-12-31'
```

3. **Checking whether it is a leap year or not**:
   To know whether the year is a leap year or not, we can use the LAST_DAY() function to check the last day of the month February of that year. If it is the 29th day then that year is leap otherwise not.
   Syntax :

```
mysql> SELECT LAST_DAY('2016-02-17');
```

Output :

```
'2016-02-29'
```

4. **Extracting the last day for the current month**:
   To know the last date of the current month ,the LAST_DAY() function is combined with the NOW() or CURDATE() function and can be executed in the following way:
   Using the NOW() function: The NOW() function in MySQL returns the current date-time stamp.
   Syntax :

```
mysql> SELECT LAST_DAY(NOW());
```

Output :

```
'2017-12-31'
```

5. **Using the CURDATE() function**: The CURDATE() function in MySQL return the current date in Date format.
   Syntax :

```
mysql> SELECT LAST_DAY(CURDATE());
```

Output :

```
'2017-12-31'
```

6. **Extracting the last day of the next month**:
   To know the last date of the next month,the last_day() function can be executed in the following way:
   Syntax :

```
mysql> SELECT LAST_DAY(CURDATE() + INTERVAL 1 MONTH);
```

Output :

```
'2018-01-31'
```

Last Updated : 21 Mar, 2018

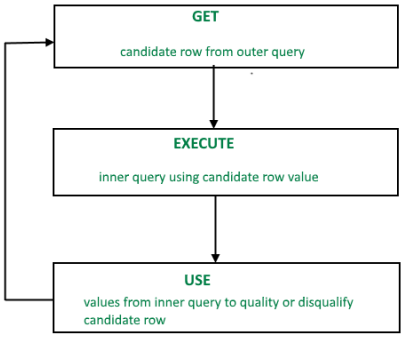
Similar Reads

1. PLSQL | LAST_DAY Function
2. LOCALTIME() and LAST_DAY() Function in MariaDB
3. MySQL | Common MySQL Queries

SQL Correlated Subqueries

MrinalVerma

Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query.



A correlated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a **SELECT**, **UPDATE**, or **DELETE** statement.

```
SELECT column1, column2, ....
FROM table1 outer
WHERE column1 operator
                    (SELECT column1, column2
                     FROM table2
                     WHERE expr1 =
                               outer.expr2);
```

A correlated subquery is one way of reading every row in a table and comparing values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query. In other words, you can use a correlated subquery to answer a multipart question whose answer depends on the value in each row processed by the parent statement.

Nested Subqueries Versus Correlated Subqueries :

With a normal nested subquery, the inner **SELECT** query runs first and executes once, returning values to be used by the main query. A correlated subquery, however, executes once for each candidate row considered by the outer query. In other words, the inner query is driven by the outer query.

> ***NOTE:*** *You can also use the* ***ANY*** *and* ***ALL*** *operator in a correlated subquery.* ***EXAMPLE of Correlated Subqueries :*** *Find all the employees who earn more than the average salary in their department.*

AD

```
SELECT last_name, salary, department_id
 FROM employees outer
 WHERE salary >
                (SELECT AVG(salary)
                 FROM employees
                 WHERE department_id =
                        outer.department_id group by department_id);
```

Other use of correlation is in **UPDATE** and **DELETE**

CORRELATED UPDATE :

```
UPDATE table1 alias1
 SET column = (SELECT expression
               FROM table2 alias2
               WHERE alias1.column =
                     alias2.column);
```

Use a correlated subquery to update rows in one table based on rows from another table.

CORRELATED DELETE :

```
DELETE FROM table1 alias1
 WHERE column1 operator
               (SELECT expression
```

```
                FROM table2 alias2
                WHERE alias1.column = alias2.column);
```

Use a correlated subquery to delete rows in one table based on the rows from another table.

Using the EXISTS Operator :

The EXISTS operator tests for existence of rows in the results set of the subquery. If a subquery row value is found the condition is flagged **TRUE** and the search does not continue in the inner query, and if it is not found then the condition is flagged **FALSE** and the search continues in the inner query.

EXAMPLE of using EXIST operator :

Find employees who have at least one person reporting to them.

```
SELECT employee_id, last_name, job_id, department_id
FROM employees outer
WHERE EXISTS ( SELECT 'X'
FROM employees
WHERE manager_id =
outer.employee_id);
```

OUTPUT :

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
108	Greenberg	FI_MGR	100
114	Raphaely	PU_MAN	30
120	Weiss	ST_MAN	50
121	Fripp	ST_MAN	50
122	Kaufling	ST_MAN	50
123	Vollman	ST_MAN	50
More than 10 rows available. Increase rows selector to view more rows.			

10 rows returned in 0.05 seconds CSV Export


```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
FROM employees
```

```
WHERE department_id
= d.department_id);
```

OUTPUT :

DEPARTMENT_ID	DEPARTMENT_NAME
120	Treasury
130	Corporate Tax
140	Control And Credit
150	Shareholder Services
160	Benefits
170	Manufacturing
180	Construction
190	Contracting
200	Operations
210	IT Support
More than 10 rows available. Increase rows selector to view more rows.	

10 rows returned in 0.18 seconds CSV Export

Last Updated : 11 Dec, 2022

39

Similar Reads

1. Difference between Nested Subquery, Correlated Subquery and Join Operation
2. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
3. Configure SQL Jobs in SQL Server using T-SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between SQL and T-SQL
8. SQL Server | Convert tables in T-SQL into XML
9. SQL SERVER | Bulk insert data from csv file using T-SQL command
10. SQL - SELECT from Multiple Tables with MS SQL Server

Previous Next

How to find Nth highest salary from a table?

Structured Query Language is a computer language that we use to interact with a relational database. Finding Nth highest salary in a table is the most common question asked in interviews. Here is a way to do this task using the dense_rank() function.

Consider the following table:

CREATE TABLE:

AD

```
CREATE TABLE emp (
  emp_name VARCHAR(50),
  emp_salary DECIMAL(10,2)
);
```

Let's insert some random data with a random name and then we will look at how to calculate the nth highest emp_salary.

Query:

```
CREATE TABLE emp (
  emp_name VARCHAR(50),
  emp_salary DECIMAL(10,2)
);
INSERT INTO emp (emp_name, emp_salary) VALUES
('Shubham Thakur', 50000.00),
('Aman Chopra', 60000.50),
('Naveen Tulasi', 75000.75),
```

```
('Bhavika uppala', 45000.25),
('Nishant jain', 80000.00);
```

Output:

emp_name	emp_salary
Shubham Thakur	50000
Aman Chopra	60000.5
Naveen Tulasi	75000.75
Bhavika uppala	45000.25
Nishant jain	80000

Query:

```
select * from(
select emp_name, emp_salary, dense_rank()
over(order by emp_salary desc)r from Emp)
where r=3;
```

1. To find the 2nd highest sal set n = 2
2. To find the 3rd highest sal set n = 3 and so on.

Let's check to find 3rd highest salary:

Output:

emp_name	emp_salary	r
Aman Chopra	60000.5	3

Using DENSE_RANK

1. DENSE_RANK computes the rank of a row in an ordered group of rows and returns the rank as a NUMBER. The ranks are consecutive integers beginning with 1.
2. This function accepts arguments as any numeric data type and returns NUMBER.
3. As an analytic function, DENSE_RANK computes the rank of each row returned from a query with respect to the other rows, based on the values of the value_exprs in the order_by_clause.
4. In the above query, the rank is returned based on the sal of the employee table. In the case of a tie, it assigns equal rank to all the rows.

Alternate Solution

```
CREATE TABLE `Employee` (
`ENAME` varchar(225) COLLATE utf8_unicode_ci NOT NULL,
`SAL` bigint(20) unsigned NOT NULL,
PRIMARY KEY (`ENAME`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

To find the 6th highest salary

Query:

```
mysql> select * from ((select * from Employee
       ORDER BY `sal` DESC limit 6 ) AS T)
       ORDER BY T.`sal` ASC limit 1;
```

Alternate Use of Limit

```
select * from Employee ORDER BY `sal`
DESC limit 5,1; // will return 6th highest
```

Output:

emp_name	emp_salary	r
Aman Chopra	60000.5	3

Alternate Solution

Suppose the task is to find the employee with the Nth highest salary from the above table. We can do this as follows:

1. Find the employees with top N distinct salaries.
2. Find the lowest salary among the salaries fetched by the above query, this will give us the Nth highest salary.
3. Find the details of the employee whose salary is the lowest salary fetched by the above query.

Query:

```
SELECT * FROM Employee WHERE sal =
        (
           SELECT MIN(sal) FROM Employee
           WHERE  sal IN (
           SELECT DISTINCT TOP N
           sal FROM Employee
           ORDER BY sal DESC
))
```

The above query will fetch the details of the employee with the Nth highest salary. Let us see how:

Consider N = 4.

Starting with the most inner query, the query:

Query:

```
SELECT DISTINCT sal
FROM Employee
ORDER BY sal DESC
LIMIT 4;
```

Output:

emp_salary
80000
75000.75
60000.5
50000

Query:

```
SELECT MIN(sal) FROM Employee WHERE sal IN (
  SELECT DISTINCT sal
  FROM Employee
  ORDER BY sal DESC
  LIMIT 4
);
```

Output:

MIN(emp_salary)
50000

You can see that the above-returned result is the required 4th highest salary.

Another Solution:

Here N = nth Highest Salary eg. 3rd Highest salary: N=3.

Syntax:

SELECT ename,sal from Employee e1 where

N-1 = (SELECT COUNT(DISTINCT sal)from Employee e2 where e2.sal > e1.sal)

Using the LIMIT Clause

Syntax:

Select Salary from table_name order by Salary DESC limit n-1,1;

Here we are ordering our salaries in descending order so we will get the highest salary first and then subsequently lower salaries. The limit clause has two components, the First component is to skip a number of rows from the top and the second component is to display the number of rows we want.

Let us see with an example :

To find the 4th Highest salary query will be

Query:

```
Select emp_sal from Emp order by emp_sal DESC limit 3,1;
```

Output:

emp_salary
70000

Here we are skipping 3 rows from the Top and returning only 1 row after skipping.

You can also find names of employees having Nth Highest Salary

Syntax:

Select Emp_name from table_name where Salary =

(Select Salary from table_name order by Salary DESC limit n-1,1);

There can be another question like finding Nth Lowest Salary. In order to do that, just reverse order using ASC (if you don't specify by default column will be ordered in ascending order).

Syntax:

Select Salary from table_name order by Salary limit n-1,1;

This article is contributed by **Rishav Shandilya**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Last Updated : 06 May, 2023 80

Similar Reads

1. SQL Query to Find Monthly Salary of Employee If Annual Salary is Given
2. SQL Query to Print the Name and Salary of the Person Having Least Salary in the Department
3. SQL Query to Find the Highest Salary of Each Department
4. Displaying Department Name Having Highest Average Salary in SQL Server
5. SQL Query to find an employee whose salary is equal to or greater than a specific number
6. Finding Average Salary of Each Department in SQL Server
7. SQL Query to Find the Highest Purchase Amount Ordered by the Each Customer
8. SQL Query to Display Nth Record from Employee Table
9. How to Select All Records from One Table That Do Not Exist in Another Table in SQL?
10. SQL Query to Filter a Table using Another Table

Previous Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

SQL - Useful Functions

SQL has many built-in functions for performing processing on string or numeric data. Following is the list of all useful SQL built-in functions –

- SQL COUNT Function - The SQL COUNT aggregate function is used to count the number of rows in a database table.
- SQL MAX Function - The SQL MAX aggregate function allows us to select the highest (maximum) value for a certain column.
- SQL MIN Function - The SQL MIN aggregate function allows us to select the lowest (minimum) value for a certain column.
- SQL AVG Function - The SQL AVG aggregate function selects the average value for certain table column.
- SQL SUM Function - The SQL SUM aggregate function allows selecting the total for a numeric column.
- SQL SQRT Functions - This is used to generate a square root of a given number.
- SQL RAND Function - This is used to generate a random number using SQL command.
- SQL CONCAT Function - This is used to concatenate any string inside any SQL command.
- SQL Numeric Functions - Complete list of SQL functions required to manipulate numbers in SQL.
- SQL String Functions - Complete list of SQL functions required to manipulate strings in SQL.