Raushan Kumar

PYSPARK
REGEXP FUNCTION
PART 2
REGEXP_EXTRACT()

Raushan Kumar
https://www.linkedin.com/in/raushan-kumar-553154297/

REGEXP IN PYSPARK

In PySpark, regex functions are used to perform operations like pattern matching, extracting substrings, and replacing text within columns in DataFrames. PySpark provides several built-in functions for regex operations through the pyspark.sql.functions module, mainly used for working with columns that contain strings.

Here is a detailed explanation of some commonly used regex-related functions in PySpark:

regexp_extract()

The regexp_extract() function in PySpark is used to **extract a portion of a string that matches a given regular expression pattern**. It is particularly useful when you want to extract specific information from a string, like parsing text, extracting parts of URLs, or extracting fields from structured text.

Syntax:

pyspark.sql.functions.regexp_extract(col: Column, pattern: str, idx: int)

- **col**: The column containing the string on which the regular expression operation will be applied.
- **pattern**: The regular expression pattern to match.
- **idx**: The index of the capturing group to extract. A capturing group is a part of the pattern enclosed in parentheses (). Index 0 returns the entire match, 1 returns the first captured group, 2 returns the second captured group, etc.

Return Type:

- **Column**: A new column with the extracted substring that matches the pattern.

Key Points to Understand:

- **Capturing Groups**: The function relies on capturing groups in regular expressions, which are parts of the regex pattern enclosed in parentheses ().
- **Indexing**: The idx argument refers to the index of the capturing group you want to extract:
  - 0: Extracts the entire match.
  - 1: Extracts the first capturing group.
  - 2: Extracts the second capturing group, and so on.

Ex1: Extract the Domain from an Email Address

Let's say you have a DataFrame with email addresses, and you want to extract the domain part (the part after @).

Example Code:

```
from pyspark.sql.functions import regexp_extract

# Create a Spark session

spark =
SparkSession.builder.master("local[1]").appName("Regexp
Extract Example").getOrCreate()

# Sample data

data = [("alice@example.com",), ("bob@domain.com",),
("charlie@xyz.com",)]

df = spark.createDataFrame(data, ["email"])

# Extract domain from email

df1=(df.withColumn('domain',

   regexp_extract(col('email'),r'@([a-zA-Z0-9._-]+)',1)))

df1.show(truncate=False)
```

Output:

```
+-----------------+------------+
|email            |domain      |
+-----------------+------------+
|alice@example.com|@example.com|
|bob@domain.com   |@domain.com |
|charlie@xyz.com  |@xyz.com    |
+-----------------+------------+
```

Explanation:

- **Regex Pattern**: r"@([a-zA-Z0-9.-]+)"
  - @: Matches the literal @ character.
  - ([a-zA-Z0-9.-]+): Matches one or more alphanumeric characters, dots, or hyphens, capturing the domain name.
- **idx=1**: The first capturing group (the part after @).

The regexp_extract() function extracts the domain part after the @ symbol for each email address.

Ex2: Extract Date from a Text

Suppose you have text entries that contain a date in the format YYYY-MM-DD, and you want to extract just the date portion.

Example Code:

```
from pyspark.sql import SparkSession

from pyspark.sql.functions import regexp_extract

# Create a Spark session

spark =
SparkSession.builder.master("local[1]").appName("Regexp
Extract Example").getOrCreate()

# Sample data

data = [("The event is on 2025-03-05.",), ("Meeting date:
2023-08-21.",), ("No date here.",)]

df = spark.createDataFrame(data, ["text"])

# Extract date in YYYY-MM-DD format

df = df.withColumn("date", regexp_extract("text", r"\d{4}-
\d{2}-\d{2}",0))

df.show(truncate=False)
```

Output:

```
+--------------------------+----------+
|text                      |date      |
+--------------------------+----------+
|The event is on 2025-03-05.|2025-03-05|
|Meeting date: 2023-08-21.  |2023-08-21|
|No date here.              |          |
+--------------------------+----------+
```

Explanation:

- **Regex Pattern**: r"\d{4}-\d{2}-\d{2}"
  - \d{4}: Matches exactly four digits (year).
  - -: Matches the hyphen.
  - \d{2}: Matches exactly two digits (month and day).
  - This pattern matches any string that looks like YYYY-MM-DD.
- **idx=0**: Extracts the entire date match.

If a date is found in the text, it is extracted, otherwise, the result is null.

Conclusion:

The regexp_extract() function in PySpark is a powerful tool for extracting specific patterns from text data. Some key use cases include:

- Extracting parts of a URL, email, or phone number.
- Extracting date or time from text.
- Parsing structured or semi-structured text.

The function uses regular expressions to match patterns in a string, and the idx parameter lets you specify which part of the match (capturing group) to extract. This makes regexp_extract() highly flexible for a variety of text extraction tasks.

By: Raushan Kumar

Please follow for more such content:

https://www.linkedin.com/in/raushan-kumar-553154297/