Master PySpark: From Zero to Big Data Hero!!

Date Function in Dataframe – Part 2

In PySpark, handling dates with the correct format and extracting date/time components such as year, month, day, etc., can be done with functions like to_date, to_timestamp, year, month, dayofmonth, hour, minute, and second. Below is a detailed explanation of how to work with date formats and extract date components.

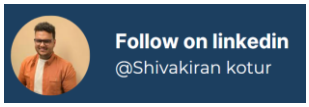
Code Explanation with Notes

1. **Default Date Parsing (to_date)**:
    - When using to_date(), the default date format is yyyy-MM-dd.
    - If the format of the string does not match this, PySpark returns null for invalid date parsing.

```
# Example: "2016-20-12" is invalid (20 is not a valid month), returns null
dateDF.select(
    to_date(lit("2016-20-12")).alias("incorrect_date"),
    to_date(lit("2017-12-11")).alias("correct_date")
).show(1)
```

```
▸ (2) Spark Jobs
+--------------+------------+
|incorrect_date|correct_date|
+--------------+------------+
|          null|  2017-12-11|
+--------------+------------+
only showing top 1 row
```

2. **Handling Custom Date Formats**:
    - You can specify a custom date format using the to_date function by providing a format string, such as yyyy-dd-MM.
    - This allows PySpark to correctly parse the dates that deviate from the default format.



```
from pyspark.sql.functions import to_date

dateFormat = "yyyy-dd-MM"
cleanDateDF = spark.range(1).select(
    to_date(lit("2017-12-11"), dateFormat).alias("correct_format_date"),
    to_date(lit("2017-20-12"), dateFormat).alias("incorrect_format_date")
)
cleanDateDF.show()
```

▸ (1) Spark Jobs

▾ cleanDateDF: pyspark.sql.dataframe.DataFrame
correct_format_date: date
incorrect_format_date: date

```
+-------------------+---------------------+
|correct_format_date|incorrect_format_date|
+-------------------+---------------------+
|         2017-11-12|           2017-12-20|
+-------------------+---------------------+
```

- Here, "2017-12-11" will be parsed correctly since it fits yyyy-dd-MM, but "2017-20-12" will return null since the day (20) is out of the valid range for December (month 12).

3. Handling Timestamps:

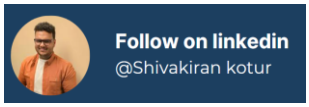
- You can use to_timestamp to convert strings with both date and time into a timestamp format. This is useful when working with datetime values.
- After casting to a timestamp, you can extract various date/time components such as the year, month, day, hour, minute, and second.

```
from pyspark.sql.functions import to_timestamp, year, month, dayofmonth, hour, minute, second, col

# Select all components (year, month, day, hour, minute, second) in a single show
cleanDateDF.select(
    to_timestamp(col("correct_format_date"), dateFormat).alias("timestamp"),
    year(to_timestamp(col("correct_format_date"), dateFormat)).alias("year"),
    month(to_timestamp(col("correct_format_date"), dateFormat)).alias("month"),
    dayofmonth(to_timestamp(col("correct_format_date"), dateFormat)).alias("day"),
    hour(to_timestamp(col("correct_format_date"), dateFormat)).alias("hour"),
    minute(to_timestamp(col("correct_format_date"), dateFormat)).alias("minute"),
    second(to_timestamp(col("correct_format_date"), dateFormat)).alias("second")
).show()
```

▸ (1) Spark Jobs

```
+-------------------+----+-----+---+----+------+------+
|          timestamp|year|month|day|hour|minute|second|
+-------------------+----+-----+---+----+------+------+
|2017-11-12 00:00:00|2017|   11| 12|   0|     0|     0|
+-------------------+----+-----+---+----+------+------+
```



Detailed Explanation of Each Function

1. **to_date**:
    - Converts a string column to a date column based on the given format. If the format does not match, null is returned.
2. **to_timestamp**:
    - Converts a string column with date and time information into a timestamp, which includes both date and time.
3. **Extracting Date Components**:
    - **year**: Extracts the year from a date or timestamp.
    - **month**: Extracts the month from a date or timestamp.
    - **dayofmonth**: Extracts the day of the month from a date or timestamp.
    - **hour**: Extracts the hour from a timestamp.
    - **minute**: Extracts the minute from a timestamp.
    - **second**: Extracts the second from a timestamp.

Sample Output

For the input "2017-12-11" (with the format yyyy-dd-MM), you can expect the following results:

- **Year**: 2017
- **Month**: 12
- **Day**: 11
- **Hour**: 0 (since no time is provided)
- **Minute**: 0
- **Second**: 0

For invalid date strings (like "2017-20-12"), you will get null in the resulting DataFrame.