How to Round a Date to the First Day of the Month in PySpark

Authored by

stats writer

February 3, 2026

To accurately normalize time-series data in PySpark, a common requirement is rounding a date field to the first day of its respective month. This vital transformation is efficiently achieved using the specialized function `date_trunc`, specifying `"month"` as the unit.

This technique effectively truncates the given date, ensuring all dates within the same calendar month map consistently to the first day of that period. This approach is fundamental for robust data analysis, particularly when aggregating metrics or performing time-series grouping, as it provides a standardized and organized representation of chronological data within a large-scale dataset.

PySpark: Round Date to First Day of Month

The Importance of Date Normalization in Data Processing

In many real-world scenarios involving big data, date and time fields are stored with high granularity, sometimes down to the second or millisecond. While this precision is useful for certain tasks, it often complicates high-level aggregation and reporting. For instance, if analysts wish to compare monthly sales volumes, having dates scattered across various days and times makes direct comparison cumbersome without prior normalization.

Date normalization, specifically rounding to the first day of the month, serves as a powerful standardization technique. By aligning all dates to a common monthly anchor point, we can ensure that group-by operations are accurate and that the resulting summarized data reflects true monthly trends rather than daily fluctuations. This consistency is essential when working with large distributed datasets managed by frameworks like PySpark, where efficient partitioning and grouping are critical for performance.

Furthermore, standardizing dates simplifies downstream processes, such as joining multiple datasets or exporting reports. When historical data spans many years, aggregating data by the normalized month allows for cleaner visualization and easier interpretation of long-term patterns and seasonal variations in business or scientific data.

Introducing the `date_trunc` Function in PySpark

PySpark leverages powerful built-in functions, many derived from standard SQL functions, to manipulate data types efficiently. The key utility for date rounding is the `pyspark.sql.functions.date_trunc` function. This function is designed to truncate a timestamp or date to the beginning of a specified unit of time. Unlike rounding (which might go up or down), truncation always rounds down to the start of the interval.

The `date_trunc` function requires two primary arguments: the format (or unit) and the column to

be truncated. When the format argument is set to `'month'`, the function ensures that the output date maintains the original year and month, but the day component is reset to `01`. If the input column is a timestamp, the time component is also reset to `00:00:00`.

Using `date_trunc` is significantly more efficient than attempting to construct the first day of the month manually using string manipulation or complex mathematical operations on date components. As an optimized native function within PySpark, it capitalizes on Spark's distributed architecture to perform the transformation across all partitions of a DataFrame quickly and reliably, even for petabytes of data.

Detailed Syntax for Date Rounding

To implement this date rounding within a DataFrame, you must first import the necessary functions module, typically aliased as `F`. The operation then involves using the `withColumn` method to append a new column or overwrite an existing one, applying `F.trunc` (which is a common alias for `date_trunc` when working within `pyspark.sql.functions`).

The general structure for applying this logic to round dates to the first day of the month in a DataFrame involves defining the new column name, specifying the `trunc` function, referencing the existing date column, and finally passing the interval `'month'`.

You can use the following standard PySpark syntax to efficiently derive a new date column representing the start of the month:

import pyspark.sql.functions as F

#add new column that rounds date to first day of month
df_new = df.withColumn('first_day_of_month', F.trunc('date', 'month'))

This specific example utilizes the `withColumn` transformation to create a new column named `first_day_of_month`. For every row, the value in the source column (referenced here as `date`) is processed by `F.trunc` using the `'month'` unit, guaranteeing that the resulting value is the date of the first day of the month associated with the original date.

Setting Up the PySpark Environment and Sample Data

To demonstrate this capability practically, we must first initialize a PySpark `SparkSession` and generate a sample DataFrame. Our sample data will simulate sales transactions, containing a date stamp and the corresponding sales volume. This setup provides a clear context for how the date truncation affects real operational data.

The following preparation steps ensure that the execution environment is ready and that the input data structure is established with the appropriate column types required for date manipulation:

from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

#define data
data = ,
,
,
,
,
,
,
]

#define column names
columns =

#create dataframe using data and column names
df = spark.createDataFrame(data, columns)

#view dataframe
df.show()

+----------+-----+
| date|sales|
+----------+-----+
|2023-04-11| 22|
|2023-04-15| 14|
|2023-04-17| 12|
|2023-05-21| 15|
|2023-05-23| 30|
|2023-10-26| 45|
|2023-10-28| 32|
|2023-10-29| 47|
+----------+-----+

As shown, the resulting DataFrame, `df`, contains dates spanning April, May, and October of 2023. These dates are currently specific to the day of the transaction. The objective is now to transform this data such that every record is associated with its corresponding monthly starting date,

simplifying subsequent monthly aggregation tasks.

Practical Example: Applying `trunc` to a DataFrame

Now that the sample data is prepared, we apply the `trunc` function. We aim to create a new column, `first_day_of_month`, which contains the date rounded down to the first day of the month, based on the existing `date` column. This is a common requirement when consolidating daily metrics into monthly summaries for executive dashboards or long-term trend analysis.

The code below demonstrates the application of `F.trunc` and then immediately displays the resulting DataFrame, `df_new`, allowing us to inspect the transformation side-by-side with the original data:

import pyspark.sql.functions as F

#add new column that rounds date to first day of month
df_new = df.withColumn('first_day_of_month', F.trunc('date', 'month'))

#view new DataFrame
df_new.show()

+----------+-----+------------------+
| date|sales|first_day_of_month|
+----------+-----+------------------+
|2023-04-11| 22| 2023-04-01|
|2023-04-15| 14| 2023-04-01|
|2023-04-17| 12| 2023-04-01|
|2023-05-21| 15| 2023-05-01|
|2023-05-23| 30| 2023-05-01|
|2023-10-26| 45| 2023-10-01|
|2023-10-28| 32| 2023-10-01|
|2023-10-29| 47| 2023-10-01|
+----------+-----+------------------+

Verifying the Output and Understanding the Results

The output `df_new` clearly shows the successful application of the truncation logic. We observe a new column, `first_day_of_month`, appended to the DataFrame. Every date in this new column retains the original year and month information but is standardized to the first day of that month (`01`).

This result confirms the reliability of using `F.trunc('date', 'month')` for date normalization in PySpark. For instance, regardless of whether a transaction occurred early, mid, or late in April 2023, the normalized date is uniformly `2023-04-01`. This makes the data immediately ready for monthly grouping operations, such as calculating the total sales per month.

Specific examples illustrating this transformation include:

The date **2023-04-11**, which is the eleventh day of April, has been correctly rounded down to the first day of that month: **2023-04-01**.
The date **2023-04-15**, which is the fifteenth day of April, has similarly been rounded to **2023-04-01**.
The date **2023-04-17**, representing the seventeenth day of April, has also been successfully normalized to **2023-04-01**.
Dates from other months, such as those in May (e.g., 2023-05-23), are mapped precisely to their respective first day (**2023-05-01**).

This method is highly scalable and ensures that chronological grouping is always consistent across all records processed by the cluster.

Conclusion: Streamlining Time-Series Data

The technique of using the `trunc` function in PySpark to round dates to the first day of the month is an essential skill for any data engineer or analyst working with time-series data. It provides a clean, native, and highly efficient way to standardize date fields, which is a prerequisite for accurate aggregation and comparative reporting.

By relying on built-in functions like `date_trunc`, we avoid the overhead of complex user-defined functions (UDFs) or inefficient string manipulations, ensuring the highest performance levels attainable within the Apache Spark ecosystem.

For further exploration into PySpark's robust date and time functionalities, analysts might also investigate related functions like `add_months`, `date_add`, and `months_between`, which offer additional capabilities for complex chronological calculations beyond simple truncation.

The following tutorials explain how to perform other common tasks in PySpark: