Date & Timestamp Functions in PySpark

We'll cover every **important date and timestamp operation**, including **functions, examples, use-cases**, and best practices.

🔷 📌 1. Getting Current Date & Timestamp

🔹 Function: `current_date()`

Returns the current date in **YYYY-MM-DD** format.

```
from pyspark.sql.functions import current_date

df = spark.range(1).withColumn("current_date", current_date())
df.show()
```

Output:

```
+-- -+- -- --- ---
--       -+       |
id|current_date|
+-- -+- -- --- ---
-- -+ | 0| 2025-
08-07 | +-- -+- --
--- --- -- -+
```

🔹 Function: `current_timestamp()`

Returns the current timestamp (date + time in `yyyy-MM-dd HH:mm:ss.SSS` format).

```
from pyspark.sql.functions import current_timestamp

df = spark.range(1).withColumn("current_ts", current_timestamp())
df.show(truncate=False)
```

Use-Case: Useful for logging ingestion time, audit columns.

🔷 📌 2. Convert String to Date/Timestamp

🔹 Function: `to_date()`

Convertsstringto `DateType`.

```
from pyspark.sql.functions import to_date

data = [("07-08-2025",)]
df = spark.createDataFrame(data, ["raw_date"])
df = df.withColumn("parsed_date", to_date("raw_date", "dd-MM-yyyy"))
df.show()
```

🔹 Function: `to_timestamp()`

Convertsstringto `TimestampType`.

```
from pyspark.sql.functions import to_timestamp

data = [("07-08-2025 09:45:00",)]
df = spark.createDataFrame(data, ["raw_ts"])
df = df.withColumn("parsed_ts", to_timestamp("raw_ts", "dd-MM-yyyy
HH:mm:ss"))
df.show(truncate=False)
```

Tip: Always specify the format when using `to_date` or `to_timestamp`.

🔷 📌 3. Extract Parts of Date/Timestamp

Use these to get **year, month, day, etc.**

```
from pyspark.sql.functions import year, month, dayofmonth, dayofweek,
dayofyear, hour, minute

data = [("2025-08-07 12:34:56",)]
df = spark.createDataFrame(data, ["ts"]).withColumn("ts",
to_timestamp("ts"))

df = df.withColumn("year", year("ts")) \


       .withColumn("month", month("ts")) \
       .withColumn("day", dayofmonth("ts")) \
       .withColumn("dow", dayofweek("ts")) \
       .withColumn("doy", dayofyear("ts")) \
       .withColumn("hour", hour("ts")) \
       .withColumn("minute", minute("ts"))

df.show()
```

Output:

ts	year	month	day	dow	doy	hour	minute
2025-08-07 12:34:56	2025	8	7	5	219	12	34

🔷 📌 4. Date Arithmetic

🔹 Add or Subtract Days: `date_add`, `date_sub`

```
from pyspark.sql.functions import date_add, date_sub

df = spark.range(1).withColumn("today", current_date())
df = df.withColumn("next_week", date_add("today", 7))
df = df.withColumn("last_week", date_sub("today", 7))
df.show()
```

🔹 Add Months: `add_months()`

```
from pyspark.sql.functions import add_months

df = df.withColumn("next_month", add_months("today", 1))
df.show()
```

Use-Case: Billing cycles, subscription renewals.

🔷 📌 5. Difference Between Dates

🔹 Function: `datediff()`

Returns number of days between two dates.

```
from pyspark.sql.functions import datediff

data = [("2025-08-01", "2025-08-07")]
df = spark.createDataFrame(data, ["start", "end"])
df = df.withColumn("diff_days", datediff("end", "start"))
```

```
df.show()
```

🔹 Function: `months_between()`

Returns the number of months between two dates.

```
from pyspark.sql.functions import months_between

df = df.withColumn("months_diff", months_between("end", "start"))
df.show()
```

Output could be decimal — e.g., 0.1935 (6 days ≈ 0.2 months).

🔷 📌 6. Formatting Dates

🔹 Function: `date_format()`

Formats date or timestamp into a custom string pattern.

```
from pyspark.sql.functions import date_format

data = [("2025-08-07 14:55:00",)]
df = spark.createDataFrame(data, ["ts"])
df = df.withColumn("ts", to_timestamp("ts"))
df = df.withColumn("formatted", date_format("ts", "dd-MMM-yyyy hh:mm
a"))
df.show(truncate=False)
```

Output: `07-Aug-2025 02:55 PM`

🔷 📌 7. Truncating Dates

🔹 Function: `trunc()`

Truncate to beginning of month/year.

```
from pyspark.sql.functions import trunc

data = [("2025-08-07",)]
df = spark.createDataFrame(data, ["dt"]).withColumn("dt",
to_date("dt"))
df = df.withColumn("month_start", trunc("dt", "MM")) \

       .withColumn("year_start", trunc("dt", "YYYY"))
df.show()
```

Output:

- `202 5-0 8-`→ Start of month
- `01 202 5-0`→ Start of year
  `1- 01`

🔷 📌 8. Working with Time Intervals

🔹 Function: `from_unixtime()` and `unix_timestamp()`

Convert between UNIX timestamp and datetime.

```
from pyspark.sql.functions import unix_timestamp, from_unixtime

df = spark.createDataFrame([("2025-08-07 14:00:00",)], ["ts"])
df = df.withColumn("ts", to_timestamp("ts"))
df = df.withColumn("unix", unix_timestamp("ts")) \

       .withColumn("back_to_ts", from_unixtime("unix"))
df.show()
```

🔷 📌 9. Filtering by Date or Time

🔹 Example: Filter last 7 days of data

```
from pyspark.sql.functions import current_date, date_sub

df.filter(df["order_date"] >= date_sub(current_date(), 7))
```

🔷 📌 10. Handling Nulls in Date Columns

```
df. fil te r(d f[" ti mes ta mp_ co l"] .i
sNo        tNu         l        l()        )
df.fillna({"timestamp_col": "2000-01-01"})
```

Always handle nulls before transformations like `to_date()`or `datediff()`.

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/