[Функции даты и отметки времени Spark SQL](https://sparkbyexamples.com/spark/spark-sql-date-and-time-functions/)

Spark SQL предоставляет встроенные стандартные функции даты и времени (включая дату и время), определенные в DataFrame API, они пригодятся, когда нам нужно выполнять операции с датой и временем. Все они принимают входные данные в виде типа даты, типа отметки времени или строки. Если это строка, она должна быть в формате, который может быть преобразован к дате, например yyyy-MM-dd и метка времени в  
 yyyy-MM-dd HH:mm:ss.SSSS и возвращает дату и метку времени соответственно; также возвращает значение NULL, если входные данные представляли собой строку, которую невозможно привести к дате и метке времени.

По возможности старайтесь использовать стандартную библиотеку, поскольку она немного более безопасна во время компиляции, обрабатывает нулевые значения и работает лучше по сравнению с [Spark UDF](https://sparkbyexamples.com/spark/spark-sql-udf/). Если ваше приложение имеет [решающее значение для производительности](https://sparkbyexamples.com/spark/spark-performance-tuning/), постарайтесь любой ценой избегать использования пользовательских UDF, поскольку они не гарантируют производительность.

Для удобства чтения я сгруппировал функции Date и Timestamp следующим образом.

1. [Функции даты Spark SQL](#_Функции_даты_Spark)
2. [Функции временных меток Spark SQL](#_Функции_временных_меток)
3. [Дата и время оконные функции](#_Дата_и_время)

Прежде чем использовать приведенные ниже примеры, убедитесь, что вы создали сессию Spark и импортировали функции SQL.

*import org.apache.spark.sql.SparkSession*

*val spark:SparkSession = SparkSession.builder()*

*.master("local[3]")*

*.appName("SparkByExample")*

*.getOrCreate()*

*spark.sparkContext.setLogLevel("ERROR")*

*import spark.sqlContext.implicits.\_*

*import org.apache.spark.sql.functions.\_*

# Функции даты Spark SQL

Нажмите на каждую ссылку в таблице ниже, чтобы получить дополнительные объяснения и рабочие примеры в Scala.

|  |  |
| --- | --- |
| ФУНКЦИЯ ДАТЫ | ФУНКЦИЯ ДАТЫ ОПИСАНИЕ |
| current\_date () : Column | Возвращает текущую дату в виде столбца даты. |
| date\_format(dateExpr: Column, format: String): Column | Преобразует дату/метку времени/строку в значение строки в формате, заданном форматом даты, заданным вторым аргументом. |
| to\_date(e: Column): Column | Преобразует столбец в *DateType* путем приведения правил к *DateType*. |
| to\_date(e: Column, fmt: String): Column | Преобразует столбец в *DateType* указанного формата. |
| add\_months(startDate: Column, numMonths: Int): Column | Возвращает дату, которая находится через numMonths после startDate. |
| date\_add(start: Column, days: Int): Column  date\_sub(start: Column, days: Int): Column | Возвращает дату, которая находится через несколько дней после начала (*start*) |
| datediff(end: Column, start: Column): Column | Возвращает количество дней от начала (*start*) до конца (*end*). |
| months\_between(end: Column, start: Column): Column | Возвращает количество месяцев между датами начала (*start*) и окончания (*end*). Возвращается целое число, если оба входных параметра имеют один и тот же день месяца или оба являются последним днем ​​соответствующего месяца. В противном случае разница рассчитывается исходя из 31 дня в месяце. |
| months\_between(end: Column, start: Column, roundOff: Boolean): Column | Возвращает количество месяцев между датами окончания (*end*) и начала (*start*). Если для параметра roundOff установлено значение true, результат округляется до 8 цифр; иначе оно не округляется. |
| next\_day(date: Column, dayOfWeek: String): Column | Возвращает первую дату, которая позже значения столбца даты, приходящегося на указанный день недели. Например, *next\_day('2015-07-27', "Воскресенье")* возвращает 2015-08-02, поскольку это первое воскресенье после 2015-07-27. |
| trunc(date: Column, format: String): Column | Возвращает дату, усеченную до единицы, указанной в формате. Например, trunc(*"2018-11-19 12:01:19", "* *year "*) возвращает 2018-01-01.  формат: “year”, “yyyy”, “yy” для усечения по году,  “month”, “mon”, “mm”’ для усечения по месяцам  “day”, “dd” для усечения по дням Другие варианты: “second”, “minute”, “hour”, “week”, “month”, “quarter” |
| year(e: Column): Column | Извлекает год как целое число из заданной даты/временной метки/строки. |
| quarter(e: Column): Column | Извлекает квартал как целое число из заданной даты/временной метки/строки. |
| month(e: Column): Column | Извлекает месяц как целое число из заданной даты/временной метки/строки. |
| dayofweek(e: Column): Column | Извлекает день недели как целое число из заданной даты/временной метки/строки. Диапазон значений от 1 для воскресенья до 7 для субботы. |
| dayofmonth(e: Column): Column | Извлекает день месяца как целое число из заданной даты/временной метки/строки. |
| dayofyear(e: Column): Column | Извлекает день года как целое число из заданной даты/временной метки/строки. |
| weekofyear(e: Column): Column | Извлекает номер недели как целое число из заданной даты/временной метки/строки. Считается, что неделя начинается в понедельник, а неделя 1 — это первая неделя, в которой более 3 дней, как определено ISO 8601. |
| last\_day(e: Column): Column | Возвращает последний день месяца, к которому принадлежит данная дата. Например, ввод «2015-07-27» возвращает «2015-07-31», поскольку 31 июля — последний день месяца в июле 2015 года. |
| from\_unixtime(ut: Column): Column | Преобразует количество секунд от эпохи Unix (1970-01-01 00:00:00 UTC) в строку, представляющую временную метку этого момента в текущем часовом поясе системы в формате yyyy-MM-dd HH:mm:ss. |
| from\_unixtime(ut: Column, f: String): Column | Преобразует количество секунд из эпохи Unix (1970-01-01 00:00:00 UTC) в строку, представляющую временную метку этого момента в текущем системном часовом поясе в заданном формате. |
| unix\_timestamp(): Column | Возвращает текущую временную метку Unix (в секундах) в виде длинного целого |
| unix\_timestamp(s: Column): Column | Преобразует строку времени в формате yyyy-MM-dd HH:mm:ss в метку времени Unix (в секундах), используя часовой пояс по умолчанию и языковой стандарт по умолчанию. |
| unix\_timestamp(s: Column, p: String): Column | Преобразует строку времени по заданному шаблону в метку времени Unix (в секундах). |

# Функции временных меток Spark SQL

Ниже приведены некоторые функции Spark SQL Timestamp. Эти функции работают как со значениями даты, так и со значениями метки времени. Выберите каждую ссылку, чтобы просмотреть описание и пример каждой функции.

Формат метки времени Spark по умолчанию - *yyyy-MM-dd HH:mm:ss.SSSS*

|  |  |
| --- | --- |
| ФУНКЦИЯ ВРЕМЕНИ | ФУНКЦИЯ ВРЕМЕНИ - ОПИСАНИЕ |
| current\_timestamp () : Column | Возвращает текущую метку времени в виде столбца метки времени. |
| current\_timestamp () : Column | Возвращает текущую метку времени в виде столбца метки времени. |
| hour(e: Column): Column | Извлекает часы как целое число из заданной даты/временной метки/строки. |
| minute(e: Column): Column | Извлекает минуты как целое число из заданной даты/временной метки/строки. |
| second(e: Column): Column | Извлекает секунды как целое число из заданной даты/временной метки/строки. |
| to\_timestamp(s: Column): Column | Преобразуется в метку времени путем приведения правил к *TimestampType*. |
| to\_timestamp(s: Column, fmt: String): Column | Converts time string with the given pattern to timestamp. |

# Дата и время оконные функции

Ниже приведены оконные функции даты и времени.

|  |  |
| --- | --- |
| ФУНКЦИЯ | ОПИСАНИЕ |
| window(timeColumn: Column, windowDuration: String,  slideDuration: String, startTime: String): Column | Группируйте строки в одно или несколько временных окон с учетом столбца, определяющего временную метку. Начало окна включает в себя, но конец окна является исключительным, например. 12:05 будет в окне [12:05,12:10), но не в [12:00,12:05). Windows может поддерживать точность до микросекунды. Windows в порядке месяцев не поддерживается. |
| window(timeColumn: Column, windowDuration: String, slideDuration: String): Column | Группируйте строки в одно или несколько временных окон с учетом столбца, определяющего временную метку. Начало окна включает в себя, но конец окна является исключительным, например. 12:05 будет в окне [12:05,12:10), но не в [12:00,12:05). Windows может поддерживать точность до микросекунды. Windows в порядке месяцев не поддерживается. Окна начинаются с 01.01.1970, 00:00:00 UTC. |
| window(timeColumn: Column, windowDuration: String): Column | Генерирует чередующиеся временные окна с учетом столбца, определяющего временную метку. Начало окна включает в себя, но конец окна является исключительным, например. 12:05 будет в окне [12:05,12:10), но не в [12:00,12:05). Windows может поддерживать точность до микросекунды. Windows в порядке месяцев не поддерживается. Окна начинаются с 01.01.1970, 00:00:00 UTC. |

# Примеры функций Spark Date

Ниже приведены наиболее часто используемые примеры функций даты.

## current\_date() and date\_format()

Мы увидим, как получить текущую дату и преобразовать дату в определенный формат даты с помощью date\_format() на примере Scala. В приведенном ниже примере дата анализируется и преобразуется из формата «yyyy-dd-mm» в формат «MM-dd-yyyy».

*import org.apache.spark.sql.functions.\_  
Seq(("2019-01-23"))  
 .toDF("Input")  
 .select(   
 current\_date()as("current\_date"),  
 col("Input"),  
 date\_format(col("Input"), "MM-dd-yyyy").as("format")  
 )  
.show()*

+-----------------+---------------+---------------------+   
|current\_date| Input |format |  
+-----------------+---------------+----------------------+   
| 2019-07-23 |2019-01-23| 01-23-2019 |  
 +------------+----------+-----------+