We have the following String functions in Apache Pig.

|  |  |
| --- | --- |
| **S.N.** | **Functions & Description** |
| 1 | [ENDSWITH(string, testAgainst)](https://www.tutorialspoint.com/apache_pig/apache_pig_endswith.htm)  To verify whether a given string ends with a particular substring. |
| 2 | [STARTSWITH(string, substring)](https://www.tutorialspoint.com/apache_pig/apache_pig_startswith.htm)  Accepts two string parameters and verifies whether the first string starts with the second. |
| 3 | [SUBSTRING(string, startIndex, stopIndex)](https://www.tutorialspoint.com/apache_pig/apache_pig_substring.htm)  Returns a substring from a given string. |
| 4 | [EqualsIgnoreCase(string1, string2)](https://www.tutorialspoint.com/apache_pig/apache_pig_equalsignorecase.htm)  To compare two stings ignoring the case. |
| 5 | [INDEXOF(string, ‘character’, startIndex)](https://www.tutorialspoint.com/apache_pig/apache_pig_indexof.htm)  Returns the first occurrence of a character in a string, searching forward from a start index. |
| 6 | [LAST\_INDEX\_OF(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_last_index_of.htm)  Returns the index of the last occurrence of a character in a string, searching backward from a start index. |
| 7 | [LCFIRST(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_lcfirst.htm)  Converts the first character in a string to lower case. |
| 8 | [UCFIRST(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_ucfirst.htm)  Returns a string with the first character converted to upper case. |
| 9 | [UPPER(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_upper.htm)  UPPER(expression) Returns a string converted to upper case. |
| 10 | [LOWER(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_lower.htm)  Converts all characters in a string to lower case. |
| 11 | [REPLACE(string, ‘oldChar’, ‘newChar’);](https://www.tutorialspoint.com/apache_pig/apache_pig_replace.htm)  To replace existing characters in a string with new characters. |
| 12 | [STRSPLIT(string, regex, limit)](https://www.tutorialspoint.com/apache_pig/apache_pig_strsplit.htm)  To split a string around matches of a given regular expression. |
| 13 | [STRSPLITTOBAG(string, regex, limit)](https://www.tutorialspoint.com/apache_pig/apache_pig_strsplittobag.htm)  Similar to the **STRSPLIT()** function, it splits the string by given delimiter and returns the result in a bag. |
| 14 | [TRIM(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_trim.htm)  Returns a copy of a string with leading and trailing whitespaces removed. |
| 15 | [LTRIM(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_ltrim.htm)  Returns a copy of a string with leading whitespaces removed. |
| 16 | [RTRIM(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_rtrim.htm)  Returns a copy of a string with trailing whitespaces removed. |

Apache Pig provides the following Date and Time functions −

|  |  |
| --- | --- |
| **S.N.** | **Functions & Description** |
| 1 | [ToDate(milliseconds)](https://www.tutorialspoint.com/apache_pig/apache_pig_todate.htm)  This function returns a date-time object according to the given parameters. The other alternative for this function are ToDate(iosstring), ToDate(userstring, format), ToDate(userstring, format, timezone) |
| 2 | [CurrentTime()](https://www.tutorialspoint.com/apache_pig/apache_pig_currenttime.htm)  returns the date-time object of the current time. |
| 3 | [GetDay(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getday.htm)  Returns the day of a month from the date-time object. |
| 4 | [GetHour(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_gethour.htm)  Returns the hour of a day from the date-time object. |
| 5 | [GetMilliSecond(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getmillisecond.htm)  Returns the millisecond of a second from the date-time object. |
| 6 | [GetMinute(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getminute.htm)  Returns the minute of an hour from the date-time object. |
| 7 | [GetMonth(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getmonth.htm)  Returns the month of a year from the date-time object. |
| 8 | [GetSecond(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getsecond.htm)  Returns the second of a minute from the date-time object. |
| 9 | [GetWeek(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getweek.htm)  Returns the week of a year from the date-time object. |
| 10 | [GetWeekYear(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getweekyear.htm)  Returns the week year from the date-time object. |
| 11 | [GetYear(datetime)](https://www.tutorialspoint.com/apache_pig/apache_pig_getyear.htm)  Returns the year from the date-time object. |
| 12 | [AddDuration(datetime, duration)](https://www.tutorialspoint.com/apache_pig/apache_pig_addduration.htm)  Returns the result of a date-time object along with the duration object. |
| 13 | [SubtractDuration(datetime, duration)](https://www.tutorialspoint.com/apache_pig/apache_pig_subtractduration.htm)  Subtracts the Duration object from the Date-Time object and returns the result. |
| 14 | [DaysBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_daysbetween.htm)  Returns the number of days between the two date-time objects. |
| 15 | [HoursBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_hoursbetween.htm)  Returns the number of hours between two date-time objects. |
| 16 | [MilliSecondsBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_millisecondsbetween.htm)  Returns the number of milliseconds between two date-time objects. |
| 17 | [MinutesBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_minutesbetween.htm)  Returns the number of minutes between two date-time objects. |
| 18 | [MonthsBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_monthsbetween.htm)  Returns the number of months between two date-time objects. |
| 19 | [SecondsBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_secondsbetween.htm)  Returns the number of seconds between two date-time objects. |
| 20 | [WeeksBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_weeksbetween.htm)  Returns the number of weeks between two date-time objects. |
| 21 | [YearsBetween(datetime1, datetime2)](https://www.tutorialspoint.com/apache_pig/apache_pig_yearsbetween.htm)  Returns the number of years between two date-time objects. |

We have the following Math functions in Apache Pig −

|  |  |
| --- | --- |
| **S.N.** | **Functions & Description** |
| 1 | [ABS(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_abs.htm)  To get the absolute value of an expression. |
| 2 | [ACOS(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_acos.htm)  To get the arc cosine of an expression. |
| 3 | [ASIN(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_asin.htm)  To get the arc sine of an expression. |
| 4 | [ATAN(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_atan.htm)  This function is used to get the arc tangent of an expression. |
| 5 | [CBRT(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_cbrt.htm)  This function is used to get the cube root of an expression. |
| 6 | [CEIL(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_ceil.htm)  This function is used to get the value of an expression rounded up to the nearest integer. |
| 7 | [COS(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_cos.htm)  This function is used to get the trigonometric cosine of an expression. |
| 8 | [COSH(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_cosh.htm)  This function is used to get the hyperbolic cosine of an expression. |
| 9 | [EXP(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_exp.htm)  This function is used to get the Euler’s number e raised to the power of x. |
| 10 | [FLOOR(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_floor.htm)  To get the value of an expression rounded down to the nearest integer. |
| 11 | [LOG(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_log.htm)  To get the natural logarithm (base e) of an expression. |
| 12 | [LOG10(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_log10.htm)  To get the base 10 logarithm of an expression. |
| 13 | [RANDOM( )](https://www.tutorialspoint.com/apache_pig/apache_pig_random.htm)  To get a pseudo random number (type double) greater than or equal to 0.0 and less than 1.0. |
| 14 | [ROUND(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_round.htm)  To get the value of an expression rounded to an integer (if the result type is float) or rounded to a long (if the result type is double). |
| 15 | [SIN(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_sin.htm)  To get the sine of an expression. |
| 16 | [SINH(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_sinh.htm)  To get the hyperbolic sine of an expression. |
| 17 | [SQRT(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_sqrt.htm)  To get the positive square root of an expression. |
| 18 | [TAN(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_tan.htm)  To get the trigonometric tangent of an angle. |
| 19 | [TANH(expression)](https://www.tutorialspoint.com/apache_pig/apache_pig_tanh.htm)  To get the hyperbolic tangent of an expression. |

User Defined Function

In addition to the built-in functions, Apache Pig provides extensive support for **U**ser **D**efined **F**unctions (UDF’s). Using these UDF’s, we can define our own functions and use them. The UDF support is provided in six programming languages, namely, Java, Jython, Python, JavaScript, Ruby and Groovy.

For writing UDF’s, complete support is provided in Java and limited support is provided in all the remaining languages. Using Java, you can write UDF’s involving all parts of the processing like data load/store, column transformation, and aggregation. Since Apache Pig has been written in Java, the UDF’s written using Java language work efficiently compared to other languages.

In Apache Pig, we also have a Java repository for UDF’s named **Piggybank**. Using Piggybank, we can access Java UDF’s written by other users, and contribute our own UDF’s.

Types of UDF’s in Java

While writing UDF’s using Java, we can create and use the following three types of functions −

* **Filter Functions** − The filter functions are used as conditions in filter statements. These functions accept a Pig value as input and return a Boolean value.
* **Eval Functions** − The Eval functions are used in FOREACH-GENERATE statements. These functions accept a Pig value as input and return a Pig result.
* **Algebraic Functions** − The Algebraic functions act on inner bags in a FOREACHGENERATE statement. These functions are used to perform full MapReduce operations on an inner bag.

Writing UDF’s using Java

To write a UDF using Java, we have to integrate the jar file **Pig-0.15.0.jar**. In this section, we discuss how to write a sample UDF using Eclipse. Before proceeding further, make sure you have installed Eclipse and Maven in your system.

Follow the steps given below to write a UDF function −

* Open Eclipse and create a new project (say **myproject**).
* Convert the newly created project into a Maven project.
* Copy the following content in the pom.xml. This file contains the Maven dependencies for Apache Pig and Hadoop-core jar files.

<project xmlns = "http://maven.apache.org/POM/4.0.0"

xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0http://maven.apache .org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>Pig\_Udf</groupId>

<artifactId>Pig\_Udf</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>org.apache.pig</groupId>

<artifactId>pig</artifactId>

<version>0.15.0</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-core</artifactId>

<version>0.20.2</version>

</dependency>

</dependencies>

</project>

* Save the file and refresh it. In the **Maven Dependencies** section, you can find the downloaded jar files.
* Create a new class file with name **Sample\_Eval** and copy the following content in it.

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

public class Sample\_Eval extends EvalFunc<String>{

public String exec(Tuple input) throws IOException {

if (input == null || input.size() == 0)

return null;

String str = (String)input.get(0);

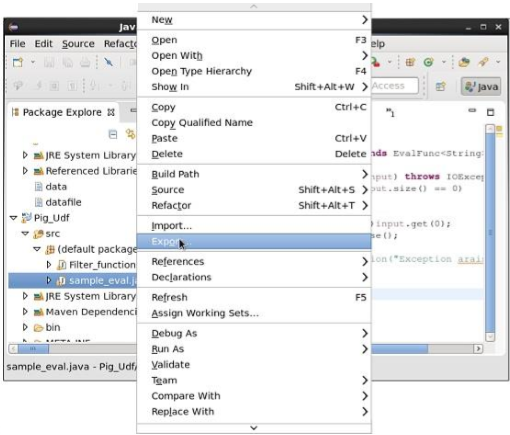
return str.toUpperCase();

}

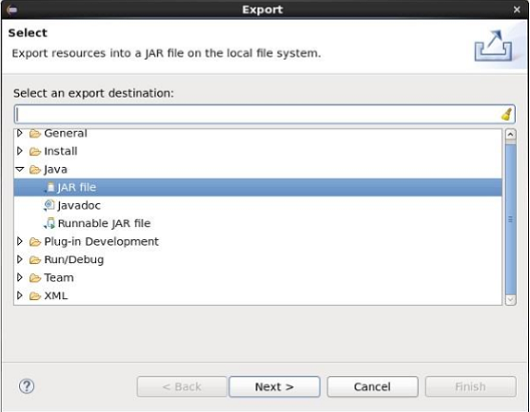
}

While writing UDF’s, it is mandatory to inherit the EvalFunc class and provide implementation to **exec()** function. Within this function, the code required for the UDF is written. In the above example, we have return the code to convert the contents of the given column to uppercase.

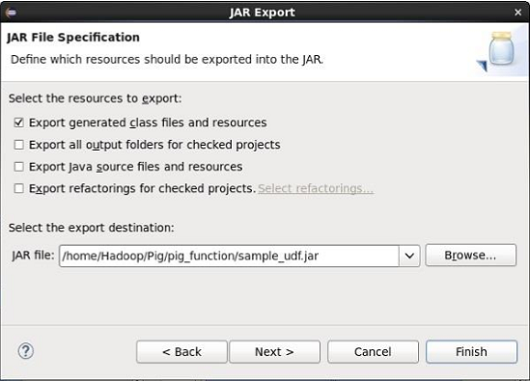
* After compiling the class without errors, right-click on the Sample\_Eval.java file. It gives you a menu. Select **export** as shown in the following screenshot.



* On clicking **export**, you will get the following window. Click on **JAR file**.



* Proceed further by clicking **Next>** button. You will get another window where you need to enter the path in the local file system, where you need to store the jar file.



* Finally click the **Finish** button. In the specified folder, a Jar file **sample\_udf.jar** is created. This jar file contains the UDF written in Java.

Using the UDF

After writing the UDF and generating the Jar file, follow the steps given below −

Step 1: Registering the Jar file

After writing UDF (in Java) we have to register the Jar file that contain the UDF using the Register operator. By registering the Jar file, users can intimate the location of the UDF to Apache Pig.

**Syntax**

Given below is the syntax of the Register operator.

REGISTER path;

**Example**

As an example let us register the sample\_udf.jar created earlier in this chapter.

Start Apache Pig in local mode and register the jar file sample\_udf.jar as shown below.

$cd PIG\_HOME/bin

$./pig –x local

REGISTER '/$PIG\_HOME/sample\_udf.jar'

**Note** − assume the Jar file in the path − /$PIG\_HOME/sample\_udf.jar

Step 2: Defining Alias

After registering the UDF we can define an alias to it using the **Define** operator.

**Syntax**

Given below is the syntax of the Define operator.

DEFINE alias {function | [`command` [input] [output] [ship] [cache] [stderr] ] };

**Example**

Define the alias for sample\_eval as shown below.

DEFINE sample\_eval sample\_eval();

Step 3: Using the UDF

After defining the alias you can use the UDF same as the built-in functions. Suppose there is a file named emp\_data in the HDFS **/Pig\_Data/** directory with the following content.

001,Robin,22,newyork

002,BOB,23,Kolkata

003,Maya,23,Tokyo

004,Sara,25,London

005,David,23,Bhuwaneshwar

006,Maggy,22,Chennai

007,Robert,22,newyork

008,Syam,23,Kolkata

009,Mary,25,Tokyo

010,Saran,25,London

011,Stacy,25,Bhuwaneshwar

012,Kelly,22,Chennai

And assume we have loaded this file into Pig as shown below.

grunt> emp\_data = LOAD 'hdfs://localhost:9000/pig\_data/emp1.txt' USING PigStorage(',')

as (id:int, name:chararray, age:int, city:chararray);

Let us now convert the names of the employees in to upper case using the UDF **sample\_eval**.

grunt> Upper\_case = FOREACH emp\_data GENERATE sample\_eval(name);

Verify the contents of the relation **Upper\_case** as shown below.

**grunt> Dump Upper\_case;**

(ROBIN)

(BOB)

(MAYA)

(SARA)

(DAVID)

(MAGGY)

(ROBERT)

(SYAM)

(MARY)

(SARAN)

(STACY)

(KELLY)