Name : Akhil Dudhatra

Div : E

Roll.no : 09

Enrollment No : 23004401110060

Day\_7

## HDFS Commands: -ls, -ls -R, -mkdir, -put, -get

### Create a file “Sample” in a local file system and export it to the HDFS File System. hdfs dfs -copyFromLocal

d:\Sample.txt /Test

hdfs dfs -put d:\Sample.txt /Test

-

### Write the HDFS command for copying a “Sample” file from HDFS to local File System. hdfs dfs -copyToLocal

/Test /Sample.txt D\

hdfs dfs -get /Test /Sample.txt d\

- **Write HDFS commands for creating “Test” directory in HDFS and then removing that directory.** hdfs dfs -mkdir /Test hdfs dfs -rm /Test

### - Write HDFS command to display complete list of directories and files of HDFS. hdfs dfs -ls/ hdfs dfs -ls/ Test

--

### Write HDFS command for displaying the contents of “Sample” text file in HDFS on screen.

hdfs dfs -cat /Test/Sample.txt

--

### Write HDFS command for copying an existing “Sample” file in a “Test” HDFS directory to some another HDFS directory.

hdfs dfs -appendToFile d:\Sample.txt /Test/Sample2.txt

--

**Practice HDFS command**

### Execute the HDFS command for getting the list of complete directories and files of HDFS.

hdfs dfs -ls/

hdfs dfs -ls/ Test

- **Execute the HDFS command for displaying the contents of some Xyz. text file in HDFS on screen.** hdfs dfs -copyFromLocal d:\Xyz.txt /Test hdfs dfs -put d:\Xyz.txt /Test

-

### Execute the HDFS command for copying an existing sample file in a given HDFS directory to some another HDFS directory

hdfs dfs -copyToLocal /Test /Sample.txt D\ hdfs dfs -get /Test /Sample.txt d\

Day\_8

### get the list of all the files in the HDFS root directory

Help

### Write a command to Listing all the files in HDFS.

hdfs dfs -ls/

hdfs dfs -ls/ Test hdfs dfs -dy/Test hdfs dfs -df/Test

## -

### Write a command to copy of a file “Abc.txt” from Local file System to Hadoop FS.

hdfs dfs -copyFromLocal d:\Sample.txt /Test

**Desirable Assignment**

## -

### Taking any data/file of your choice, execute the HDFS command forcopying a given sample file in local file system to HDFS.

hdfs dfs -copyFromLocal d:\Sample.txt /Test

## -

### Taking any data/file of your choice, execute the HDFS command for copying a given sample file in HDFS to local File System.

hdfs dfs -copyToLocal /Test /Sample.txt D\ hdfs dfs -get /Test /Sample.txt d\

### Execute the HDFS commands for creating some sample directory in HDFS and then removing that directory

hdfs dfs -rm /Test/sample.txt

# Day\_9

### Working with Pig Operators/Functions (LOAD, DUMP, FOREACH, GROUP, DISTINCT,LIMIT,ORDER BY) Write a pig script to load and store “Student data”. (Student file contain Roll no, Name, Marksand GPA)

**001,Ravi Solanki,51,7.9 002,Dhruv Sharma,54,7.8 003,Akhil,57,8.8**

### 004,Samir,67,9.0

**005,Jay,54,8.9**

### hadoop fs -mkdir /pig\_example

**hdfs dfs -copyFromLocal d:\student.txt /pig\_example**

### student = LOAD 'hdfs://localhost:9000/pig\_example/student.txt' USING PigStorage(',') as (roll:int,name:chararray,mark:int,CGP:float );

**dump student**

### Filter all the students who are having GPA>5Ans1 =

FILTER student BY (float)CGP>8; dump Ans1

(3,Akhil,57,8.8)

(4,Samir,67,9.0)

(5,Jay,54,8.9)

### Display the name of all Students in Uppercase.

((1,Ravi Solanki),RAVI SOLANKI) ((2,Dhruv Sharma),DHRUV SHARMA) ((3,Akhil),AKHIL)

((4,Samir),SAMIR)

((5,Jay),JAY)

### Group tuples of students based on their GPA.

group\_data\_cgp =GROUP student by CGP; dump group\_data\_cgp

(7.8,{(2,Dhruv Sharma,54,7.8)})

(7.9,{(1,Ravi Solanki,51,7.9)})

(8.8,{(3,Akhil,57,8.8)})

(8.9,{(5,Jay,54,8.9)})

(9.0,{(4,Samir,67,9.0)})

**Remove duplicates tuple of Student list**. distinct\_data = DISTINCT student;

dump distinct\_data

### Display first three tuples from “student” relation.

top\_3\_student = LIMIT student 3;

dump top\_3\_student; (1,Ravi Solanki,51,7.9) (2,Dhruv Sharma,54,7.8) (3,Akhil,57,8.8)

**Display the names of students in ascending order**. asc\_student\_by\_name = ORDER student by name ASC;dump asc\_student\_by\_name

(1,Ravi Solanki,51,7.9) (4,Samir,67,9.0)

(3,Akhil,57,8.8)

(5,Jay,54,8.9)

(2,Dhruv Sharma,54,7.8)

# Day\_9\_question\_2

(1,DDLU,1986,3.2,7560)

(2,XYZ,1985,3.8,6300)

(3,ABC,1988,4.1,7802)

(4,PQR,1993,3.7,6022)

(5,AAA,1991,3.4,5420)

(6,ZZY,2004,3.9,4904)

(7,De danadan,1987,3.4,5623) (8,GCET,1987,3.4,7563)

(9,PPP,1990,3.2,6244)

(10,PQQQ,2004,3.1,6956)

#### movie = LOAD 'hdfs://localhost:9000/movie1/MOVIE.csv'USING PigStorage(',')as(id:int,name:chararray,year:int,rating:float,duration:int); dump movie

**Write a pig script to load and store movies\_data.csv.Filter:**

**Filter movie whose rating is higher than 3.5.**

**movie\_rating = FILTER movie BY (float)rating>3.5; dump movie\_rating**

(2,XYZ,1985,3.8,6300)

(3,ABC,1988,4.1,7802)

(4,PQR,1993,3.7,6022)

(6,ZZY,2004,3.9,4904)

Store the results data from pig into new name my\_movies.

**store movie into 'my\_movies'; cat my\_movies**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | DDLU | 1986 | 3.2 | 7560 | |
| 2 | XYZ | 1985 | 3.8 | 6300 | |
| 3 | ABC | 1988 | 4.1 | 7802 | |
| 4 | PQR | 1993 | 3.7 | 6022 | |
| 5 | AAA | 1991 | 3.4 | 5420 | |
| 6 | ZZY | 2004 | 3.9 | 4904 | |
| 7 | De danadan | | 1987 | 3.4 | 5623 |
| 8 | GCET 1987 | | 3.4 | 7563 |  |
| 9 | PPP 1990 | | 3.2 | 6244 |  |
| 10 | PQQQ 2004 | | 3.1 | 6956 |  |

### Display all movie information the result.

#### dump movie

(1,DDLU,1986,3.2,7560)

(2,XYZ,1985,3.8,6300)

(3,ABC,1988,4.1,7802)

(4,PQR,1993,3.7,6022)

(5,AAA,1991,3.4,5420)

(6,ZZY,2004,3.9,4904)

(7,De danadan,1987,3.4,5623) (8,GCET,1987,3.4,7563)

(9,PPP,1990,3.2,6244)

(10,PQQQ,2004,3.1,6956)

### List the movies that were released between 1950 and 1960. YEAR = FILTER movie By year > 1990 and year<1997;

#### dump YEAR

(4,PQR,1993,3.7,6022)

(5,AAA,1991,3.4,5420)

### List the movies that start with the Alphabet F = FILTER movie by name matches 'D.\*';

#### dump F;

(1,DDLU,1986,3.2,7560)

(7,De danadan,1987,3.4,5623)

### List the movies that have duration greater than 2 hours.

#### minutesdur = FILTER movie by duration >7200; dump dur

(1,DDLU,1986,3.2,7560)

(3,ABC,1988,4.1,7802)

(8,GCET,1987,3.4,7563)

### List the movie names its duration in minutes.

#### movie\_duration = Foreach movie GENERATE name ,(double)(duration/60); dump movie\_duration

(DDLU,126.0)

(XYZ,105.0)

(ABC,130.0)

(PQR,100.0)

(AAA,90.0)

(ZZY,81.0)

(De danadan,93.0)

(GCET,126.0)

(PPP,104.0)

(PQQQ,115.0)

**Group Statement in PIG.**

### List the years and the number of movies released each year. Order by in PIGStatement.

#### group\_year = group movie by year; dump group\_year

(1985,{(2,XYZ,1985,3.8,6300)})

(1986,{(1,DDLU,1986,3.2,7560)})

(1987,{(8,GCET,1987,3.4,7563),

(7,De danadan,1987,3.4,5623)})

(1988,{(3,ABC,1988,4.1,7802)})

(1990,{(9,PPP,1990,3.2,6244)})

(1991,{(5,AAA,1991,3.4,5420)})

(1993,{(4,PQR,1993,3.7,6022)})

(2004,{(10,PQQQ,2004,3.1,6956),

(6,ZZY,2004,3.9,4904)})

**count\_by\_year = FOREACH group\_year GENERATE group,COUNT(movie);dump count\_by\_year**

### List all the movies in the ascending order of year.

**asc\_student\_by\_name = ORDER student by name ASC; dump asc\_student\_by\_name**

#### asc\_order = ORDER movie by year ASC; dump asc\_order

(2,XYZ,1985,3.8,6300)

(1,DDLU,1986,3.2,7560)

(8,GCET,1987,3.4,7563)

(7,De danadan,1987,3.4,5623) (3,ABC,1988,4.1,7802)

(9,PPP,1990,3.2,6244)

(5,AAA,1991,3.4,5420)

(4,PQR,1993,3.7,6022)

(10,PQQQ,2004,3.1,6956)

(6,ZZY,2004,3.9,4904)

(1985,1)

(1986,1)

(1987,2)

(1988,1)

(1990,1)

(1991,1)

(1993,1)

(2004,2)

### List all the movies in the descending order of year. Limit operator in pig.

#### dsc\_order = ORDER movie by year DESC; dump dsc\_order

(10,PQQQ,2004,3.1,6956)

(6,ZZY,2004,3.9,4904)

(4,PQR,1993,3.7,6022)

(5,AAA,1991,3.4,5420)

(9,PPP,1990,3.2,6244)

(3,ABC,1988,4.1,7802)

(8,GCET,1987,3.4,7563)

(7,De danadan,1987,3.4,5623) (1,DDLU,1986,3.2,7560)

(2,XYZ,1985,3.8,6300)

### Display Top 5 movies.

**top\_3\_movie = LIMIT movie 5; dump top\_3\_movie**

(1,DDLU,1986,3.2,7560)

(2,XYZ,1985,3.8,6300)

(3,ABC,1988,4.1,7802)

(4,PQR,1993,3.7,6022)

(5,AAA,1991,3.4,5420)

# DAY\_10

### Join two relations namely Student and department (Rno, DeptNo, DeptName) basedon the values contain in the roll no column.

**Merge content of two relation Student and department. Partition a relation based on the GPA’s acquired by students. To calculate the average marks for each student.**

**Calculate maximum marks of each student. Cou**nt the number of tuples in a bag

#### Assume that we have two files namely customers.txt and orders.txt in the /pig\_data/directory of HDFS as shown below

**customers.txt**

1,Ramesh,32,Ahmedabad,2000.00 2,Khilan,25,Delhi,1500.00 3,kaushik,23,Kota,2000.00 4,Chaitali,25,Mumbai,6500.00 5,Hardik,27,Bhopal,8500.00 6,Komal,22,MP,4500.00

7,Muffy,24,Indore,10000.00

#### orders.txt

102,2009-10-08 00:00:00,3,3000

100,2009-10-08 00:00:00,3,1500

101,2009-11-20 00:00:00,2,1560

103,2008-05-20 00:00:00,4,2060

#### cust1 = LOAD 'hdfs://localhost:9000/orders/customers.txt' USING PigStorage(',') as(id:int, name:chararray, age:int, city:chararray, price:int);

**dump cust1**

#### orde1= LOAD 'hdfs://localhost:9000/orders/orders.txt' USING PigStorage(',') as (o\_id:int, o\_date:chararray,cid:int,price:int);

**dump orde1 self join**

#### cust2 = LOAD 'hdfs://localhost:9000/orders/customers.txt' USING PigStorage(',') as(id:int, name:chararray, age:int,

**city:chararray, price:int); dump cust2**

(1,Ramesh,32,Ahmedabad,2000) (2,Khilan,25,Delhi,1500) (3,kaushik,23,Kota,2000) (4,Chaitali,25,Mumbai,6500) (5,Hardik,27,Bhopal,8500) (6,Komal,22,MP,4500)

(7,Muffy,24,Indore,10000)

#### cust3 = JOIN cust1 By id,cust2 by id;dump cust3

(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000) (2,Khilan,25,Delhi,1500,2,Khilan,25,Delhi,1500) (3,kaushik,23,Kota,2000,3,kaushik,23,Kota,2000) (4,Chaitali,25,Mumbai,6500,4,Chaitali,25,Mumbai,6500) (5,Hardik,27,Bhopal,8500,5,Hardik,27,Bhopal,8500) (6,Komal,22,MP,4500,6,Komal,22,MP,4500)

(7,Muffy,24,Indore,10000,7,Muffy,24,Indore,10000)

**inner join**

**customer\_orders = JOIN cust1 by id,orde1 By cid; dump customer\_orders;**

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)

(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)

(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)

(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

**left join**

**outer\_left = JOIN cust1 by id LEFT OUTER,orde1 by cid; outer\_left = JOIN orde1 by cid LEFT OUTER,cust1 by id; dump outer\_left**

(1,Ramesh,32,Ahmedabad,2000,,,,) (2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)

(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)

(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)

(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

(5,Hardik,27,Bhopal,8500,,,,)

(6,Komal,22,MP,4500,,,,)

(7,Muffy,24,Indore,10000,,,,)

#### outer\_left = JOIN orde1 by cid LEFT OUTER,cust1 by id; dump outer\_left;

(101,2009-11-20 00:00:00,2,1560,2,Khilan,25,Delhi,1500)

(100,2009-10-08 00:00:00,3,1500,3,kaushik,23,Kota,2000)

(102,2009-10-08 00:00:00,3,3000,3,kaushik,23,Kota,2000)

(103,2008-05-20 00:00:00,4,2060,4,Chaitali,25,Mumbai,6500)

#### outer\_right = JOIN cust1 by id RIGHT, orde1 by cid; dump outer\_right

(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)

(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)

(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)

(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

#### outer\_full = JOIN cust1 by id FULL OUTER, orde1 by cid; dump outer\_full

(1,Ramesh,32,Ahmedabad,2000,,,,) (2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)

(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)

(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)

(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

#### cust\_order = UNION cust1 , orde1; dump cust\_order

(1,Ramesh,32,Ahmedabad,2000) (2,Khilan,25,Delhi,1500) (3,kaushik,23,Kota,2000) (4,Chaitali,25,Mumbai,6500) (5,Hardik,27,Bhopal,8500) (6,Komal,22,MP,4500)

(7,Muffy,24,Indore,10000) (102,2009-10-08 00:00:00,3,3000)

(100,2009-10-08 00:00:00,3,1500)

(101,2009-11-20 00:00:00,2,1560)

(103,2008-05-20 00:00:00,4,2060)

#### SPLIT cust1 into Below\_23 if age<23,bet22\_25 if(age>22 and age<25); dump Below\_23;

(6,Komal,22,MP,4500)

#### dump bet22\_25;

(3,kaushik,23,Kota,2000)

(7,Muffy,24,Indore,10000)

**SPLIT**

**SPLIT cust1 into AGEBelow\_25 if age<25,AGEABOVE\_25 if(age>=25); dump AGEBelow\_25**

(3,kaushik,23,Kota,2000) (6,Komal,22,MP,4500)

(7,Muffy,24,Indore,10000)

**dump AGEABOVE\_25** (1,Ramesh,32,Ahmedabad,2000) (2,Khilan,25,Delhi,1500) (4,Chaitali,25,Mumbai,6500) (5,Hardik,27,Bhopal,8500)

**FILter :**

**cust\_city = FILTER cust1 by city =='Mumbai'; dump cust\_city**

#### (4,Chaitali,25,Mumbai,6500)

**cust\_name =FILTER cust1 by name == 'Komal'; dump cust\_name;**

(6,Komal,22,MP,4500)

#### cust\_name\_tokenize = foreach cust1 Generate TOKENIZE(name); dump cust\_name\_tokenize

({(Ramesh)})

({(Khilan)})

({(kaushik)})

({(Chaitali)})

({(Hardik)})

({(Komal)})

({(Muffy)})

cust\_name\_tokenize = foreach cust1 Generate TOKENIZE(city);dump cust\_name\_tokenize ({(Ahmedabad)})

({(Delhi)})

({(Kota)})

({(Mumbai)})

({(Bhopal)})

({(MP)})

({(Indore)})

#### tobag = FOREACH cust1 Generate TOBAG (id,name,city); dump tobag

({(1),(Ramesh),(Ahmedabad)})

({(2),(Khilan),(Delhi)})

({(3),(kaushik),(Kota)})

({(4),(Chaitali),(Mumbai)})

({(5),(Hardik),(Bhopal)})

({(6),(Komal),(MP)})

({(7),(Muffy),(Indore)})

#### totuple = FOREACH cust1 Generate TOTUPLE(id,name,city); dump totuple

((1,Ramesh,Ahmedabad)) ((2,Khilan,Delhi))

((3,kaushik,Kota)) ((4,Chaitali,Mumbai)) ((5,Hardik,Bhopal))

((6,Komal,MP))

((7,Muffy,Indore))

#### toMAP = FOREACH cust1 Generate TOMAP(name,city); dump toMAP

([Ramesh#Ahmedabad]) ([Khilan#Delhi]) ([kaushik#Kota]) ([Chaitali#Mumbai]) ([Hardik#Bhopal]) ([Komal#MP]) ([Muffy#Indore])

#### ==========top==========

**hdfs dfs -cat /orders/customers.txt**

#### hdfs dfs -copyFromLocal d:\orders.txt /orders

**hdfs dfs -mkdir /orders**

#### hdfs dfs -cat /orders/orders.txt

**hdfs dfs -copyFromLocal d:\customers.txt /orders**

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

#### hdfs dfs -cat /orders/emp\_data.txt

**emp1 = LOAD 'hdfs://localhost:9000/orders/emp\_data.txt' USING PigStorage(',') as (id:int,name:chararray, age:int, city:chararray);**

#### dump emp1

**emp\_group = Group emp1 BY age; dump emp\_group**

(22,{(12,Kelly,22,Chennai),(7,Robert,22,newyork ),(6,Maggy,22,Chennai ),(1,Robin,22,newyork )})

(23,{(8,Syam,23,Kolkata ),(5,David,23,Bhuwaneshwar ),(3,Maya,23,Tokyo ),(2,BOB,23,Kolkata )})

(25,{(11,Stacy,25,Bhuwaneshwar ),(10,Saran,25,London ),(9,Mary,25,Tokyo ),(4,Sara,25,London )})

#### data\_top = FOREACH emp\_group {top = TOP(2, 0, emp1) }; GENERATE top; Dump data\_top;

({(7,Robert,22,newyork ),(12,Kelly,22,Chennai)})

({(5,David,23,Bhuwaneshwar ),(8,Syam,23,Kolkata )}) ({(10,Saran,25,London ),(11,Stacy,25,Bhuwaneshwar )})

**functions**

**end\_endwith = FOREACH emp1 Generate (id,name),ENDSWITH (name,'n'); dump end\_endwith**

((1,Robin),true)

((2,BOB),false)

((3,Maya),false)

((4,Sara),false)

((5,David),false)

((6,Maggy),false)

((7,Robert),false)

((8,Syam),false)

((9,Mary),false)

((10,Saran),true)

((11,Stacy),false)

((12,Kelly),false)

#### startwith\_data = FOREACH emp1 Generate(id,name),STARTSWITH(name,'Ro'); dump startwith\_data;

((1,Robin),true)

((2,BOB),false)

((3,Maya),false)

((4,Sara),false)

((5,David),false)

((6,Maggy),false)

((7,Robert),true)

((8,Syam),false)

((9,Mary),false)

((10,Saran),false)

((11,Stacy),false)

((12,Kelly),false)

#### custt = Group cust1 ALL;

**avg = foreach custt GENERATE(cust1.name, cust1.age), AVG(cust1.age); dump avg;**

Total input paths to process : 1 (({(Muffy),(Komal),(Hardik),(Chaitali),(kaushik),(Khilan),(Ramesh)},{(24),(22),(27),(25),(23),(25),(32)}),25.42857142857142 7)

#### top\_3\_custt = LIMIT custt 3; dump top\_3\_cutts

(1,Ravi Solanki,51,7.9) (2,Dhruv Sharma,54,7.8) (3,Akhil,57,8.8)

# Day\_11

001,1989/09/26 09:00:00

002,1980/06/20 10:22:00

003,1990/12/19 03:11:44

**date\_data = LOAD 'hdfs://localhost:9000/emp/date.txt' USING PigStorage(',')as (id:int,date:chararray);**

ToDate(milliseconds)

#### This function returns a date-time object according to the given parameters. The otheralternative for this function are ToDate(iosstring), ToDate(userstring, format), ToDate(userstring, format, timezone)

**todate\_data = foreach date\_data generate ToDate(date,'yyyy/MM/dd HH:mm:ss')as (date\_time:DateTime >);**

**CurrentTime()**

returns the date-time object of the current time. currenttime\_data = foreach todate\_data generate CurrentTime();

### GetDay(datetime)

Returns the day of a month from the date-time object.

**getday\_data = foreach todate\_data generate(date\_time), GetDay(date\_time);**

### GetHour(datetime)

Returns the hour of a day from the date-time object.

#### gethour\_data = foreach todate\_data generate (date\_time), GetHour(date\_time);

**GetMilliSecond(datetime)** Returns the millisecond of a second from the date-time object.

**GetMinute(datetime)** Returns the minute of an hour from the date-time object.

**GetMonth(datetime)** Returns the month of a year from the date-time object. **GetSecond(datetime)** Returns the second of a minute from the date-time object. **GetWeek(datetime)** Returns the week of a year from the date-time object.

**GetWeekYear(datetime)** Returns the week year from the date-time object.

**GetYear(datetime)** Returns the year from the date-time object.

**AddDuration(datetime, duration)** Returns the result of a date-time object along with the duration object.

Note − The Duration is represented in ISO 8601 standard. According to ISO 8601 standard P is placed at the beginning, while representing the duration and it is called as duration designator.Likewise,

Y is the year designator. We use this after declaring the year.Example − P1Y represents 1 year.

M is the month designator. We use this after declaring the month.Example − P1M represents 1 month. W is the week designator. We use this after declaring the week.Example − P1W represents 1 week.

D is the day designator. We use this after declaring the day.Example − P1D represents 1 day.

T is the time designator. We use this before declaring the time.Example − PT5H represents 5 hours. H is the hour designator. We use this after declaring the hour.Example − PT1H represents 1 hour. M is the minute designator. We use this after declaring the minute.

Example − PT1M represents 1 minute.

S is the second designator. We use this after declaring the second.Example − PT1S represents 1 second.

#### date\_duration = LOAD 'hdfs://localhost:9000/emp/date.txt' USING PigStorage(',') as (id:int,date:chararray, duration:chararray)

**Add\_duration\_data = foreach date\_duration generate(date,duration),AddDuration(ToDate(date,'yyyy/MM/dd HH:mm:ss'), duration);**

### SubtractDuration(datetime, duration)

Subtracts the Duration object from the Date-Time object and returns the result.

### DaysBetween(datetime1, datetime2)

Returns the number of days between the two date-time objects.

#### doj\_dob\_data = LOAD 'hdfs://localhost:9000/pig\_data/doj\_dob.txt' USING PigStorage(',') as(id:int, dob:chararray, doj:chararray);

**daysbetween\_data = foreach doj\_dob\_data generate DaysBetween(ToDate(doj,'dd/MM/yyyy HH:mm:ss'),ToDate(dob,'dd/MM/yyyy HH:mm:ss'));**

**HoursBetween(datetime1, datetime2)** Returns the number of hours between two date-time objects.

**MilliSecondsBetween(datetime1, datetime2)** Returns the number of milliseconds between two date-time objects.

**MinutesBetween(datetime1, datetime2)** Returns the number of minutes between two date-time objects. **MonthsBetween(datetime1, datetime2)** Returns the number of months between two date-time objects. **SecondsBetween(datetime1, datetime2)** Returns the number of seconds between two date-time objects. **WeeksBetween(datetime1, datetime2)** Returns the number of weeks between two date-time objects.

**YearsBetween(datetime1, datetime2)** Returns the number of years between two date-time objects.