# Analytic functions

Analytic functions, available since Hive 0.11.0, are a special group of functions that scan the multiple input rows to compute each output value. Analytic functions are usually used with OVER, PARTITION BY, ORDER BY, and the windowing specification.

# Example

create table stocks

(date\_ String, Ticker String, Open Double, High Double, Low Double, Close Double, Volume\_for\_the\_day int)

row format delimited fields terminated by ',';

load data local inpath '' into table stocks;

# Lag

This function returns the values of the previous row. You can specify an integer offset which designates the row position else it will take the default integer offset as

select ticker,date\_,close,lag(close,1) over(partition by ticker) as yesterday\_price from stocks

using lag we can display the yesterday’s closing price of the ticker. Lag is to be used with over function, inside the over function you can use partition or order by classes.

# Lead

This function returns the values from the following rows. You can specify an integer offset which designates the row position else it will take the default integer offset as

select ticker,date\_,close,case(lead(close,1) over(partition by ticker)-close)>0 when true then "higher" when false then "lesser" end as Changes from stocks

# FIRST\_VALUE

It returns the value of the first row from that window. With the below query, you can see the first row high price of the ticker for all the days.

select ticker,first\_value(high) over(partition by ticker) as first\_high from stocks

# LAST\_VALUE

It is the reverse of FIRST\_VALUE. It returns the value of the last row from that window. With the below query, you can see the last row high price value of the ticker for all the days.

select ticker,last\_value(high) over(partition by ticker) as first\_high from stocks

# Rank

The rank function will return the rank of the values as per the result set of the over clause. If two values are same then it will give the same rank to those 2 values and then for the next value, the sub-sequent rank will be skipped.

select ticker,close,rank() over(partition by ticker order by close) as closing from stocks

# Dense\_rank

It is same as the rank() function but the difference is if any duplicate value is present then the rank will not be skipped for the subsequent rows. Each unique value will get the ranks in a sequence.

The below query will rank the closing prices of the stock for each ticker.

select ticker,close,dense\_rank() over(partition by ticker order by close) as closing from stocks

# Cume\_dist

It returns the cumulative distribution of a value. It results from 0 to 1. For suppose if the total number of records are 10 then for the 1st row the cume\_dist will be 1/10 and for the second 2/10 and so on till 10/10.

This cume\_dist will be calculated in accordance with the result set returned by the over clause. The below query will result in the cumulative of each record for every ticker.

select ticker,cume\_dist() over(partition by ticker order by close) as cummulative from stocks

# Percent\_rank

It returns the percentage rank of each row within the result set of over clause.

Percent\_rank is calculated in accordance with the rank of the row and the calculation is as follows (rank-1)/(total\_rows\_in\_group – 1).

If the result set has only one row then the percent\_rank will be 0.

select ticker,close,percent\_rank() over(partition by ticker order by close) as closing from stocks

# Ntile

It returns the bucket number of the particular value. For suppose if you say Ntile(5) then it will create 5 buckets based on the result set of the over clause after that it will place the first 20% of the records in the 1st bucket and so on till 5th bucket.

select ticker,ntile(5) over(partition by ticker order by close ) as bucket from stocks