Spark SQL - Pyspark

Notebook:	Spark		
Created:	5/9/2018 6:42 AM	**Updated:**	5/12/2018 3:44 PM
Author:	ashu41228@gmail.com		

How to Open Spark SQL

```
[dgadiraju@gw01 ~]$ spark-sql --master yarn --conf spark.ui.port=12567
```

Running SQL Queries in PYSPARK

```
>>> sqlContext.sql("select * from limit 10").show()
```

sqlContext.sql will return data frame in order to display the values we need to use show() method.

Functions Manipulating Strings:

initcap() --> to make each word's first letter in upper case
SPLIT() --> to split the string based on delimiters

```
substr or substring
instr
like
rlike
length
lcase or lower
ucase or upper
trim, ltrim, rtrim
lpad, rpad
```

```
substr(str, pos[, len]) - returns the substring of str that starts at pos and is of length len orsubstr(bin, pos[, len]) - returns
 the slice of byte array that starts at pos and is of length len
Time taken: 0.247 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select substr('Hello World, How are you', 7, 5);
OK
World
Time taken: 0.246 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select substr('Hello World, How are you', -3);
OK
you
Time taken: 0.283 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select substr('Hello World, How are you', -7, 3);
OK
```

instr() : It will print first Occurence of the start of string to be searched

```
Time taken: 0.266 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select instr('Hello World, How are you', ' ');
OK
6
Time taken: 0.229 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select instr('Hello World, How are you', 'World');
OK
7
```

LIKE-

```
Time taken: 0.24 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select "Hello World, How are you" like 'Hello%';
OK
true
```

RLIKE --> to work with regular string

length --> to work with length of string

LPAD--

```
Time taken: 0.299 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select lpad(2, 12, '0');
OK
000000000002
Time taken: 0.259 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select lpad(12, 12, '0');
OK
000000000012
```

SPLIT and INDEX

```
hive (dgadiraju_retail_db_txt)> select split("Hello World, how are you", ' ');
OK
["Hello","World,","how","are","you"]
Time taken: 0.253 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select index(split("Hello World, how are you", ' '), 4);
OK
you
Time taken: 0.412 seconds, Fetched: 1 row(s)
```

Date Manipulation-

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
Y	Week year	Year	2009; 09
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day name in week	Text	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	Number	1
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800
X	Time zone	ISO 8601 time zone	-08; -0800; -08:00

```
hive (dgadiraju_retail_db_txt)> select current_date;
OK
2017-10-07
Time taken: 0.258 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select current_timestamp;
OK
2017-10-07 20:07:36.44
Time taken: 0.339 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select date_format(current_date, 'y');
OK
2017
Time taken: 0.29 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select date_format(current_date, 'd');
OK
7
Time taken: 0.277 seconds, Fetched: 1 row(s)
hive (dgadiraju_retail_db_txt)> select date_format(current_date, 'D');
OK
280
Time taken: 0.281 seconds, Fetched: 1 row(s)
```

```
hive (dgadiraju_retail_db_txt)> select to_date(current_timestamp);
OK
2017-10-07
Time taken: 0.244 seconds, Fetched: 1 row(s)
```

```
hive (dgadiraju_retail_db_txt)> select to_unix_timestamp(current_date);
OK
1507348800
```

```
hive (dgadiraju_retail_db_txt)> select from_unixtime(1507421802);
OK
2017-10-07 20:16:42
```

```
hive (dgadiraju_retail_db_txt)> select to_date(order_date) from orders limit 10;
OK
2013-07-25
2013-07-25
2013-07-25
2013-07-25
2013-07-25
2013-07-25
2013-07-25
2013-07-25
2013-07-25
2013-07-25
Time taken: 0.191 seconds, Fetched: 10 row(s)
hive (dgadiraju_retail_db_txt)> select date_add(order_date, 10) from orders limit 10;
OK
2013-08-04
2013-08-04
2013-08-04
2013-08-04
2013-08-04
2013-08-04
2013-08-04
2013-08-04
2013-08-04
2013-08-04
```

NVL() - for handling null

```
hive (dgadiraju_retail_db_txt)> select nvl(order_status, 'Status Missing') from orders limit 100;
```

when order_status is null then it will print 'Status Missing'

Windowing and analytical function

This section introduces the Hive QL enhancements for windowing and analytics functions. See "Windowing Specifications in HQL" (attached to HIVE-4197) for details. HIVE-896 has more information, including links to earlier documentation in the initial comments.

All of the windowing and analytics functions operate as per the SQL standard.

The current release supports the following functions for windowing and analytics:

1. Windowing functions
   - LEAD
     - The number of rows to lead can optionally be specified. If the number of rows to lead is not specified, the lead is one row.
     - Returns null when the lead for the current row extends beyond the end of the window.
   - LAG
     - The number of rows to lag can optionally be specified. If the number of rows to lag is not specified, the lag is one row.
     - Returns null when the lag for the current row extends before the beginning of the window.
   - FIRST_VALUE
   - LAST_VALUE
2. The OVER clause
   - OVER with standard aggregates:
     - COUNT
     - SUM
     - MIN
     - MAX
     - AVG
   - OVER with a PARTITION BY statement with one or more partitioning columns of any primitive datatype.
   - OVER with PARTITION BY and ORDER BY with one or more partitioning and/or ordering columns of any datatype.
     - OVER with a window specification. Windows can be defined separately in a WINDOW clause. Window specifications support the following formats:

```
(ROWS | RANGE) BETWEEN (UNBOUNDED | [num]) PRECEDING AND ([num] PRECEDING | CURRENT ROW | (UNBOUNDED | [num]) F
(ROWS | RANGE) BETWEEN CURRENT ROW AND (CURRENT ROW | (UNBOUNDED | [num]) FOLLOWING)
(ROWS | RANGE) BETWEEN [num] FOLLOWING AND (UNBOUNDED | [num]) FOLLOWING
```

When ORDER BY is specified with missing WINDOW clause, the WINDOW specification defaults to `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`.

When both ORDER BY and WINDOW clauses are missing, the WINDOW specification defaults to `ROW BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`.

USE OF PARTITION FUNCTION

Before using analytical function

```
select o.order_id, o.order_date, o.order_status, round(sum(oi.order_item_subtotal), 2) order_revenue
from orders o join order_items oi
on o.order_id = oi.order_item_order_id
where o.order_status in ('COMPLETE', 'CLOSED')
group by o.order_id, o.order_date, o.order_status
having sum(oi.order_item_subtotal) >= 1000
order by o.order_date, order_revenue desc;
```

After using analytical function

```
select o.order_id, o.order_date, o.order_status, oi.order_item_subtotal,
round(sum(oi.order_item_subtotal) over (partition by o.order_id), 2) order_revenue,
oi.order_item_subtotal/round(sum(oi.order_item_subtotal) over (partition by o.order_id), 2)
from orders o join order_items oi
on o.order_id = oi.order_item_order_id
where o.order_status in ('COMPLETE', 'CLOSED')
order by o.order_date, order_revenue desc;
```

Use of Ranking,ROW_number() Function

```
select * from (
select o.order_id, o.order_date, o.order_status, oi.order_item_subtotal,
round(sum(oi.order_item_subtotal) over (partition by o.order_id), 2) order_revenue,
oi.order_item_subtotal/round(sum(oi.order_item_subtotal) over (partition by o.order_id), 2) pct_revenue,
round(avg(oi.order_item_subtotal) over (partition by o.order_id), 2) avg_revenue,
rank() over (partition by o.order_id order by oi.order_item_subtotal desc) rnk_revenue,
dense_rank() over (partition by o.order_id order by oi.order_item_subtotal desc) dense_rnk_revenue,
pct_rank() over (partition by o.order_id order by oi.order_item_subtotal desc) pct_rnk_revenue,
row_number() over (partition by o.order_id order by oi.order_item_subtotal desc) rn_orderby_revenue,
row_number() over (partition by o.order_id) rn_revenue
from orders o join order_items oi
on o.order_id = oi.order_item_order_id
where o.order_status in ('COMPLETE', 'CLOSED')) q
where order_revenue >= 1000
order by order_date, order_revenue desc, rank_revenue;
```

Use of Windowing Function

The current release supports the following functions for windowing and analytics:

1. Windowing functions
   - LEAD
     - The number of rows to lead can optionally be specified. If the number of rows to lead is not specified, the lead is one row.
     - Returns null when the lead for the current row extends beyond the end of the window.
   - LAG
     - The number of rows to lag can optionally be specified. If the number of rows to lag is not specified, the lag is one row.
     - Returns null when the lag for the current row extends before the beginning of the window.
   - FIRST_VALUE
   - LAST_VALUE