There are different interfaces to run Spark SQL queries

* Using Hive
* spark-sql
* pyspark

**Launching spark-sql**

* To launch the spark-sql use below command

spark-sql --master yarn --conf spark.ui.port=12567

* To list the database use this command **show databases**

We can use Hive DDL and DML statements to create tables and get data into tables. We need to understand

* Create table command
* Columns and Data Types
* Row delimiter for text file format
* Other file formats
* Loading or Inserting data into Hive tables

|  |
| --- |
| create database dgadiraju\_retail\_db\_txt; |
|  | use dgadiraju\_retail\_db\_txt; |
|  |  |
|  | create table orders ( |
|  | order\_id int, |
|  | order\_date string, |
|  | order\_customer\_id int, |
|  | order\_status string |
|  | ) row format delimited fields terminated by ',' |
|  | stored as textfile; |
|  |  |
|  | load data local inpath '/data/retail\_db/orders' into table orders; |
|  |  |
|  | create table order\_items ( |
|  | order\_item\_id int, |
|  | order\_item\_order\_id int, |
|  | order\_item\_product\_id int, |
|  | order\_item\_quantity int, |
|  | order\_item\_subtotal float, |
|  | order\_item\_product\_price float |
|  | ) row format delimited fields terminated by ',' |
|  | stored as textfile; |
|  |  |
|  | load data local inpath '/data/retail\_db/order\_items' into table order\_items; |

**Creating tables and insert data**

* Hive supports other file formats
  + avro
  + orc
  + parquet
  + sequencefile
  + and more
* When we use special file formats, most likely
  + Create staging table where structure matches data
  + Create table with the format you are interested in
  + Use insert command to get data from stage table into the final table

|  |
| --- |
| create database dgadiraju\_retail\_db\_orc; |
|  | use dgadiraju\_retail\_db\_orc; |
|  |  |
|  | create table orders ( |
|  | order\_id int, |
|  | order\_date string, |
|  | order\_customer\_id int, |
|  | order\_status string |
|  | ) stored as orc; |
|  |  |
|  | insert into table orders select \* from dgadiraju\_retail\_db\_txt.orders; |
|  |  |
|  | create table order\_items ( |
|  | order\_item\_id int, |
|  | order\_item\_order\_id int, |
|  | order\_item\_product\_id int, |
|  | order\_item\_quantity int, |
|  | order\_item\_subtotal float, |
|  | order\_item\_product\_price float |
|  | ) stored as orc; |
|  |  |
|  | insert into table order\_items select \* from dgadiraju\_retail\_db\_txt.order\_items; |

All hive commands or queries can be submitted or executed using SQLContext’s sql API as part of PySpark

**Running Hive Queries**

* Filtering (horizontal and vertical)
* Functions
* Row-level transformations
* Joins
* Aggregation
* Sorting
* Set Operations
* Analytical Functions
* Windowing Functions

Spark SQL or Hive have bunch of functions to apply transformations on the data.

* String manipulating functions
* Data manipulating functions
* Typecast functions
* Functions to deal with nulls
* Aggregating functions
* Conditional expressions
* and more

String Manipulation  
Understanding string manipulation APIs helps us processing data as part of lambda or anonymous functions used in Spark APIs

Extracting data – split and get required fields  
Converting data types – type cast functions  
Discarding unnecessary columns  
Derive new expressions with data from different fields  
create table customers (  
customer\_id int,  
customer\_fname varchar(45),  
customer\_lname varchar(45),  
customer\_email varchar(45),  
customer\_password varchar(45),  
customer\_street varchar(255),  
customer\_city varchar(45),  
customer\_state varchar(45),  
customer\_zipcode varchar(45)  
) row format delimited fields terminated by ‘,’  
stored as textfile;  
load data local inpath ‘/data/retail\_db/customers’ into table customers;

spark-sql - <property>

<name>hive.metastore.warehouse.dir</name>

<value>/apps/hive/warehouse</value>

</property>

-master yarn --conf spark.ui.port=12567

set hive.metastore.warehouse.dir;

hive.metastore.warehouse.dir=/apps/hive/warehouse

set hive.cli.print.header=true;

show databases like "saurabhbhanwala\*"

> ;

OK

pyspark --master yarn --conf spark.ui.port=12544 --executor-memory 2G --num-executors 1

>>> sqlContext.sql( "show databases like 'saurabhbhanwala\*'")

DataFrame[result: string]

>>> sqlContext.sql( "show databases like 'saurabhbhanwala\*'").show()

+--------------------+

| result|

+--------------------+

| saurabhbhanwala|

|saurabhbhanwala\_r...|

|saurabhbhanwala\_r...|

+--------------------+

>>> sqlContext.sql("show tables").show()

+-----------+-----------+

| tableName|isTemporary|

+-----------+-----------+

|order\_items| false|

| orders| false|

+-----------+-----------+

sqlContext.sql("describe formatted orders").show()

+--------------------+

| result|

+--------------------+

|# col\_name ...|

| |

|order\_id ...|

|order\_date ...|

|order\_customer\_id...|

|order\_status ...|

| |

|# Detailed Table ...|

|Database: ...|

|Owner: ...|

|CreateTime: ...|

|LastAccessTime: ...|

|Protect Mode: ...|

|Retention: ...|

|Location: ...|

|Table Type: ...|

|Table Parameters:...|

| numFiles ...|

| numRows ...|

| rawDataSize ...|

+--------------------+

only showing top 20 rows

>>> for i in sqlContext.sql("describe formatted orders").collect(): print(i)

...

Row(result=u'# col\_name \tdata\_type \tcomment ')

Row(result=u'\t \t ')

Row(result=u'order\_id \tint \t ')

Row(result=u'order\_date \tstring \t ')

Row(result=u'order\_customer\_id \tint \t ')

Row(result=u'order\_status \tstring \t ')

Row(result=u'\t \t ')

Row(result=u'# Detailed Table Information\t \t ')

Row(result=u'Database: \tsaurabhbhanwala\_retail\_db\_text\t ')

Row(result=u'Owner: \tsaurabhbhanwala \t ')

Row(result=u'CreateTime: \tMon Aug 31 04:28:56 EDT 2020\t ')

Row(result=u'LastAccessTime: \tUNKNOWN \t ')

Row(result=u'Protect Mode: \tNone \t ')

Row(result=u'Retention: \t0 \t ')

Row(result=u'Location: \thdfs://nn01.itversity.com:8020/apps/hive/warehouse/saurabhbhanwala\_retail\_db\_text.db/orders\t ')

Row(result=u'Table Type: \tMANAGED\_TABLE \t ')

Row(result=u'Table Parameters:\t \t ')

Row(result=u'\tnumFiles \t1 ')

Row(result=u'\tnumRows \t0 ')

Row(result=u'\trawDataSize \t0 ')

Row(result=u'\ttotalSize \t2999944 ')

Row(result=u'\ttransient\_lastDdlTime\t1598863044 ')

Row(result=u'\t \t ')

Row(result=u'# Storage Information\t \t ')

Row(result=u'SerDe Library: \torg.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe\t ')

Row(result=u'InputFormat: \torg.apache.hadoop.mapred.TextInputFormat\t ')

Row(result=u'OutputFormat: \torg.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat\t ')

Row(result=u'Compressed: \tNo \t ')

Row(result=u'Num Buckets: \t-1 \t ')

Row(result=u'Bucket Columns: \t[] \t ')

Row(result=u'Sort Columns: \t[] \t ')

Row(result=u'Storage Desc Params:\t \t ')

Row(result=u'\tfield.delim \t, ')

Row(result=u'\tserialization.format\t, ')

sqlContext.sql("select \* from orders limit 10").show()

+--------+--------------------+-----------------+---------------+

|order\_id| order\_date|order\_customer\_id| order\_status|

+--------+--------------------+-----------------+---------------+

| 1|2013-07-25 00:00:...| 11599| CLOSED|

| 2|2013-07-25 00:00:...| 256|PENDING\_PAYMENT|

| 3|2013-07-25 00:00:...| 12111| COMPLETE|

| 4|2013-07-25 00:00:...| 8827| CLOSED|

| 5|2013-07-25 00:00:...| 11318| COMPLETE|

| 6|2013-07-25 00:00:...| 7130| COMPLETE|

| 7|2013-07-25 00:00:...| 4530| COMPLETE|

| 8|2013-07-25 00:00:...| 2911| PROCESSING|

| 9|2013-07-25 00:00:...| 5657|PENDING\_PAYMENT|

| 10|2013-07-25 00:00:...| 5648|PENDING\_PAYMENT|

+--------+--------------------+-----------------+---------------+

describe function length;

OK

tab\_name

length(str | binary) - Returns the length of str or number of bytes in binary data

Time taken: 0.1 seconds, Fetched: 1 row(s)

hive (default)> select length('Hello World');

OK

\_c0

11

select length(order\_status) from orders limit 10;

OK

\_c0

6

15

8

6

8

8

8

10

15

15

Time taken: 0.241 seconds, Fetched: 10 row(s)

String functions::

Substr or substring

Intstr

Like

Rlike

Lcase or lower

Upper or ucase

Length

Trim.ltrim,rtrim

Lpad,rpad

initcap

select substr('Hello World, How are you', 14)

> ;

OK

\_c0

How are you

select substr('Hello World, How are you', 7, 5)

> ;

OK

\_c0

hive (saurabhbhanwala\_retail\_db\_text)> select substr('Hello World, How are you', -3);

OK

\_c0

you

hive (saurabhbhanwala\_retail\_db\_text)> select substr('Hello World, How are you', -7,3);

OK

\_c0

are

hive (saurabhbhanwala\_retail\_db\_text)> select instr('Hello World, How are you', ' ');

OK

\_c0

6

hive (default)> select instr('Hello World, How are you', 'World');

OK

\_c0

7

describe function like;

OK

tab\_name

like(str, pattern) - Checks if str matches pattern

Time taken: 0.286 seconds, Fetched: 1 row(s)

hive (default)>

hive (default)> select 'Hello World, how r u' like 'How';

OK

\_c0

false

Time taken: 0.469 seconds, Fetched: 1 row(s)

hive (default)> select 'Hello World, how r u' like 'Hello%';

OK

\_c0

true

hive (default)> select length('Hello World');

OK

\_c0

11

Time taken: 0.482 seconds, Fetched: 1 row(s)

hive (default)> select lower('Hello World');

OK

\_c0

hello world

Time taken: 0.333 seconds, Fetched: 1 row(s)

hive (default)> select upper('Hello World');

OK

\_c0

HELLO WORLD

Time taken: 0.381 seconds, Fetched: 1 row(s)

hive (default)>

> select ucase('Hello World');

OK

\_c0

HELLO WORLD

> describe function initcap;

OK

tab\_name

initcap(str) - Returns str, with the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by white space.

Time taken: 0.089 seconds, Fetched: 1 row(s)

select trim(' hello world '), length(trim(' hello world '));

OK

\_c0 \_c1

hello world 11

Time taken: 0.368 seconds, Fetched: 1 row(s)

hive (default)> describe function lpad;

OK

tab\_name

lpad(str, len, pad) - Returns str, left-padded with pad to a length of len

Time taken: 0.067 seconds, Fetched: 1 row(s)

hive (default)> select lpad(2, 2, '0');

OK

\_c0

02

select cast(12 as int);

OK

\_c0

12

Time taken: 0.382 seconds, Fetched: 1 row(s)

hive (default)>

==

use saurabhbhanwala\_retail\_db\_text;

OK

Time taken: 0.539 seconds

hive (saurabhbhanwala\_retail\_db\_text)> select order\_date from orders limit 10;

OK

order\_date

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

2013-07-25 00:00:00.0

Time taken: 1.304 seconds, Fetched: 10 row(s)

==

(saurabhbhanwala\_retail\_db\_text)> select cast(substr(order\_date, 6, 2) as int) from orders limit 10;

OK

\_c0

7

7

7

7

7

7

7

7

7

7

select split("Hello World, How are you", ' ')

> ;

OK

\_c0

["Hello","World,","How","are","you"]

select index(split("Hello World, How are you", ' '),4);

OK

\_c0

you

==

Current\_date

Current\_timestamp

Date\_add

Date\_format

Date\_sub

Datediff

Day

Dayofmonth

To\_date

To\_unix\_timestamp

To\_utc\_timestamp

From\_unixtime

From\_utc\_timestamp

Minute

Month

Month\_between

Next\_day

select current\_date;

OK

\_c0

2020-09-01

select current\_timestamp;

OK

\_c0

2020-09-01 05:30:25.189

Time taken: 0.427 seconds, Fetched: 1 row(s)

select date\_format(current\_date, 'y');

OK

\_c0

2020

select day(current\_date);

OK

\_c0

1

select to\_date(current\_timestamp);

OK

\_c0

2020-09-01

select to\_unix\_timestamp(current\_date);

OK

\_c0

1598932800

select to\_unix\_timestamp(current\_timestamp);

OK

\_c0

1598953662

select from\_unixtime(1598953662);

OK

\_c0

2020-09-01 05:47:42

select to\_date(from\_unixtime(1598953662));

OK

\_c0

2020-09-01

Time taken: 0.331 seconds, Fetched: 1 row(s)

hive (saurabhbhanwala\_retail\_db\_text)>

aggregate functions

select count(1) from orders;

Query ID = saurabhbhanwala\_20200901060408\_997685dd-8392-4a45-ab62-b62aa85e4760

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41268, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41268/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41268

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-01 06:04:51,191 Stage-1 map = 0%, reduce = 0%

2020-09-01 06:04:56,505 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.85 sec

2020-09-01 06:05:01,723 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.51 sec

MapReduce Total cumulative CPU time: 5 seconds 510 msec

Ended Job = job\_1589064448439\_41268

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.51 sec HDFS Read: 3008185 HDFS Write: 6 SUCCESS

Total MapReduce CPU Time Spent: 5 seconds 510 msec

OK

\_c0

68883

Time taken: 57.38 seconds, Fetched: 1 row(s)

select sum(order\_item\_subtotal) from orders;

FAILED: SemanticException [Error 10004]: Line 1:11 Invalid table alias or column reference 'order\_item\_subtotal': (possible column names are: order\_id, order\_date, order\_customer\_id, order\_status)

hive (saurabhbhanwala\_retail\_db\_text)> select sum(order\_item\_subtotal) from orders\_items;

FAILED: SemanticException [Error 10001]: Line 1:37 Table not found 'orders\_items'

hive (saurabhbhanwala\_retail\_db\_text)> select sum(order\_item\_subtotal) from order\_Items;

Query ID = saurabhbhanwala\_20200901061407\_369f2ae0-6d69-47e2-8824-37593fa7e03c

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41270, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41270/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41270

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-01 06:14:16,285 Stage-1 map = 0%, reduce = 0%

2020-09-01 06:14:22,508 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.09 sec

2020-09-01 06:14:28,721 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.97 sec

MapReduce Total cumulative CPU time: 6 seconds 970 msec

Ended Job = job\_1589064448439\_41270

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.97 sec HDFS Read: 5418369 HDFS Write: 20 SUCCESS

Total MapReduce CPU Time Spent: 6 seconds 970 msec

OK

\_c0

3.432262059842491E7

Time taken: 23.759 seconds, Fetched: 1 row(s)

select count(1), count(distinct order\_status) from orders;

Query ID = saurabhbhanwala\_20200901061826\_1b8faeaa-21f6-4820-b989-c99bc34371ee

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41271, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41271/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41271

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-01 06:18:38,033 Stage-1 map = 0%, reduce = 0%

2020-09-01 06:18:44,257 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.72 sec

2020-09-01 06:18:49,871 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.41 sec

MapReduce Total cumulative CPU time: 5 seconds 410 msec

Ended Job = job\_1589064448439\_41271

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.41 sec HDFS Read: 3008892 HDFS Write: 8 SUCCESS

Total MapReduce CPU Time Spent: 5 seconds 410 msec

OK

c0 c1

68883 9

Case nvl

describe function case;

OK

tab\_name

CASE a WHEN b THEN c [WHEN d THEN e]\* [ELSE f] END - When a = b, returns c; when a = d, return e; else return f

Time taken: 0.063 seconds, Fetched: 1 row(s)

select distinct order\_status from orders;

Query ID = saurabhbhanwala\_20200901080633\_b3ac6ee5-c267-4795-bb88-359161da47e5

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41287, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41287/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41287

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-01 08:06:48,031 Stage-1 map = 0%, reduce = 0%

2020-09-01 08:06:53,320 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.18 sec

2020-09-01 08:06:59,563 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.78 sec

MapReduce Total cumulative CPU time: 5 seconds 780 msec

Ended Job = job\_1589064448439\_41287

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.78 sec HDFS Read: 3007835 HDFS Write: 99 SUCCESS

Total MapReduce CPU Time Spent: 5 seconds 780 msec

OK

order\_status

CANCELED

CLOSED

COMPLETE

ON\_HOLD

PAYMENT\_REVIEW

PENDING

PENDING\_PAYMENT

PROCESSING

SUSPECTED\_FRAUD

Time taken: 27.191 seconds, Fetched: 9 row(s)

select case order\_status when 'CLOSED' then 'No Action'

> when 'COMPLETE' then 'No Action'

> from orders limit 10;

FAILED: ParseException line 3:0 missing KW\_END at 'from' near '<EOF>'

hive (saurabhbhanwala\_retail\_db\_text)> select case order\_status when 'CLOSED' then 'No Action'

> when 'COMPLETE' then 'No Action' end

> from orders limit 10;

OK

\_c0

No Action

NULL

No Action

No Action

No Action

No Action

No Action

NULL

NULL

NULL

Time taken: 0.352 seconds, Fetched: 10 row(s)

select order\_status,case order\_status

when 'CLOSED' then 'No action'

when 'COMPLETE' then 'No Action'

when 'ON\_HOLD' then 'Pending Action'

when 'PAYMENT\_REVIEW' then 'Pending Action'

when 'PENDING' then 'Pending Action'

when 'PENDING\_PAYMENT' then ‘Pending Action’

else 'Risky' end from orders limit 10;

|  |
| --- |
| select order\_status, |
|  | case |
|  | when order\_status IN ('CLOSED', 'COMPLETE') then 'No Action' |
|  | when order\_status IN ('ON\_HOLD', 'PAYMENT\_REVIEW', 'PENDING', 'PENDING\_PAYMENT', 'PROCESSING') then 'Pending Action' |
|  | else 'Risky' |
|  | end from orders limit 10; |

Let us explore how we can perform row-level transformations. First, let us check typical scenarios for row-level transformations.

* Data cleansing – removing special characters
* Standardization – eg: phone number, we might want to get phone numbers from different sources and it might be represented the different manner in different systems. When we get onto downstream systems we have to represent phone number in one standard format.
* Discarding or filtering out unnecessary data
* Unpivoting the data, one row with many columns might have to return a collection of rows

select concat(substr(order\_date, 1,4), substr(order\_date, 6, 2)) from orders limit 10;

OK

\_c0

201307

201307

201307

201307

201307

201307

201307

201307

201307

201307

select date\_format('2020-09-03' , 'YYYMM');

OK

\_c0

202009

Time taken: 0.324 seconds, Fetched: 1 row(s)

hive (saurabhbhanwala\_retail\_db\_text)>

select cast(date\_format('2020-09-03', 'YYYYMM') as int);

OK

\_c0

202009

Time taken: 0.363 seconds, Fetched: 1 row(s)

Joining two tables

select count(1) from orders o inner join customers c on o.order\_customer\_id = c.customer\_id;

Query ID = saurabhbhanwala\_20200903010027\_2d66e321-0df4-4124-9e2d-921379b99ce0

Total jobs = 1

Execution log at: /tmp/saurabhbhanwala/saurabhbhanwala\_20200903010027\_2d66e321-0df4-4124-9e2d-921379b99ce0.log

2020-09-03 01:00:37 Starting to launch local task to process map join; maximum memory = 1046478848

2020-09-03 01:00:41 Dump the side-table for tag: 1 with group count: 12435 into file: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-00-27\_240\_8167960219452000239-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile01--.hashtable

2020-09-03 01:00:41 Uploaded 1 File to: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-00-27\_240\_8167960219452000239-1/-local-10004/HashTable-Stage-2/MapJoin-mapfile01--.hashtable (249791 bytes)

2020-09-03 01:00:41 End of local task; Time Taken: 3.237 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 1 out of 1

Number of reduce tasks determined at compile time: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41758, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41758/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41758

Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1

2020-09-03 01:00:52,774 Stage-2 map = 0%, reduce = 0%

2020-09-03 01:00:59,057 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 4.67 sec

2020-09-03 01:01:05,303 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 7.29 sec

MapReduce Total cumulative CPU time: 7 seconds 290 msec

Ended Job = job\_1589064448439\_41758

MapReduce Jobs Launched:

Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 7.29 sec HDFS Read: 3012709 HDFS Write: 6 SUCCESS

Total MapReduce CPU Time Spent: 7 seconds 290 msec

OK

\_c0

68883

select o.\*, c.\* from orders o inner join customers c on o.order\_customer\_id = c.customer\_id limit 10;

Query ID = saurabhbhanwala\_20200903010224\_72838d5f-c353-41a1-ae46-00b8834a92c2

Total jobs = 1

Execution log at: /tmp/saurabhbhanwala/saurabhbhanwala\_20200903010224\_72838d5f-c353-41a1-ae46-00b8834a92c2.log

2020-09-03 01:02:28 Starting to launch local task to process map join; maximum memory = 1046478848

2020-09-03 01:02:29 Dump the side-table for tag: 1 with group count: 12435 into file: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-02-24\_656\_8128053488183708541-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable

2020-09-03 01:02:29 Uploaded 1 File to: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-02-24\_656\_8128053488183708541-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile11--.hashtable (1156846 bytes)

2020-09-03 01:02:29 End of local task; Time Taken: 1.799 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 1 out of 1

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job\_1589064448439\_41759, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41759/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41759

Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0

2020-09-03 01:02:38,785 Stage-3 map = 0%, reduce = 0%

2020-09-03 01:02:46,048 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 4.37 sec

MapReduce Total cumulative CPU time: 4 seconds 370 msec

Ended Job = job\_1589064448439\_41759

MapReduce Jobs Launched:

Stage-Stage-3: Map: 1 Cumulative CPU: 4.37 sec HDFS Read: 141992 HDFS Write: 1171 SUCCESS

Total MapReduce CPU Time Spent: 4 seconds 370 msec

OK

o.order\_id o.order\_date o.order\_customer\_id o.order\_status c.customer\_id c.customer\_fname c.customer\_lname c.customer\_email c.customer\_password c.customer\_street c.customer\_city c.customer\_state c.customer\_zipcode

1 2013-07-25 00:00:00.0 11599 CLOSED 11599 Mary Malone XXXXXXXXX XXXXXXXXX 8708 Indian Horse Highway Hickory NC 28601

2 2013-07-25 00:00:00.0 256 PENDING\_PAYMENT 256 David Rodriguez XXXXXXXXX XXXXXXXXX 7605 Tawny Horse Falls Chicago IL 60625

3 2013-07-25 00:00:00.0 12111 COMPLETE 12111 Amber Franco XXXXXXXXX XXXXXXXXX 8766 Clear Prairie Line Santa Cruz CA 95060

4 2013-07-25 00:00:00.0 8827 CLOSED 8827 Brian Wilson XXXXXXXXX XXXXXXXXX 8396 High Corners San Antonio TX 78240

5 2013-07-25 00:00:00.0 11318 COMPLETE 11318 Mary Henry XXXXXXXXX XXXXXXXXX 3047 Silent Embers Maze Caguas PR 00725

6 2013-07-25 00:00:00.0 7130 COMPLETE 7130 Alice Smith XXXXXXXXX XXXXXXXXX 8852 Iron Port Brooklyn NY 11237

7 2013-07-25 00:00:00.0 4530 COMPLETE 4530 Mary Smith XXXXXXXXX XXXXXXXXX 1073 Green Leaf Green Miami FL 33161

8 2013-07-25 00:00:00.0 2911 PROCESSING 2911 Mary Smith XXXXXXXXX XXXXXXXXX 9166 Golden Nectar Corner Caguas PR 00725

9 2013-07-25 00:00:00.0 5657 PENDING\_PAYMENT 5657 Mary James XXXXXXXXX XXXXXXXXX 1389 Dusty Circuit Lakewood OH 44107

10 2013-07-25 00:00:00.0 5648 PENDING\_PAYMENT 5648 Joshua Smith XXXXXXXXX XXXXXXXXX 864 Iron Spring Stead Memphis TN 38111

Time taken: 22.618 seconds, Fetched: 10 row(s)

select o.\*, c.\* from orders o left outer join customers c on o.order\_customer\_id = c.customer\_id limit 10;

Query ID = saurabhbhanwala\_20200903010406\_a3841dac-8646-459c-ab51-057ca1d198e8

Total jobs = 1

Execution log at: /tmp/saurabhbhanwala/saurabhbhanwala\_20200903010406\_a3841dac-8646-459c-ab51-057ca1d198e8.log

2020-09-03 01:04:10 Starting to launch local task to process map join; maximum memory = 1046478848

2020-09-03 01:04:11 Dump the side-table for tag: 1 with group count: 12435 into file: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-04-06\_375\_6609586976546451410-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile21--.hashtable

2020-09-03 01:04:11 Uploaded 1 File to: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-04-06\_375\_6609586976546451410-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile21--.hashtable (1156846 bytes)

2020-09-03 01:04:11 End of local task; Time Taken: 1.602 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 1 out of 1

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job\_1589064448439\_41760, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41760/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41760

Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0

2020-09-03 01:04:21,074 Stage-3 map = 0%, reduce = 0%

2020-09-03 01:04:26,328 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 3.04 sec

MapReduce Total cumulative CPU time: 3 seconds 40 msec

Ended Job = job\_1589064448439\_41760

MapReduce Jobs Launched:

Stage-Stage-3: Map: 1 Cumulative CPU: 3.04 sec HDFS Read: 141776 HDFS Write: 1171 SUCCESS

Total MapReduce CPU Time Spent: 3 seconds 40 msec

OK

o.order\_id o.order\_date o.order\_customer\_id o.order\_status c.customer\_id c.customer\_fname c.customer\_lname c.customer\_email c.customer\_password c.customer\_street c.customer\_city c.customer\_state c.customer\_zipcode

1 2013-07-25 00:00:00.0 11599 CLOSED 11599 Mary Malone XXXXXXXXX XXXXXXXXX 8708 Indian Horse Highway Hickory NC 28601

2 2013-07-25 00:00:00.0 256 PENDING\_PAYMENT 256 David Rodriguez XXXXXXXXX XXXXXXXXX 7605 Tawny Horse Falls Chicago IL 60625

3 2013-07-25 00:00:00.0 12111 COMPLETE 12111 Amber Franco XXXXXXXXX XXXXXXXXX 8766 Clear Prairie Line Santa Cruz CA 95060

4 2013-07-25 00:00:00.0 8827 CLOSED 8827 Brian Wilson XXXXXXXXX XXXXXXXXX 8396 High Corners San Antonio TX 78240

5 2013-07-25 00:00:00.0 11318 COMPLETE 11318 Mary Henry XXXXXXXXX XXXXXXXXX 3047 Silent Embers Maze Caguas PR 00725

6 2013-07-25 00:00:00.0 7130 COMPLETE 7130 Alice Smith XXXXXXXXX XXXXXXXXX 8852 Iron Port Brooklyn NY 11237

7 2013-07-25 00:00:00.0 4530 COMPLETE 4530 Mary Smith XXXXXXXXX XXXXXXXXX 1073 Green Leaf Green Miami FL 33161

8 2013-07-25 00:00:00.0 2911 PROCESSING 2911 Mary Smith XXXXXXXXX XXXXXXXXX 9166 Golden Nectar Corner Caguas PR 00725

9 2013-07-25 00:00:00.0 5657 PENDING\_PAYMENT 5657 Mary James XXXXXXXXX XXXXXXXXX 1389 Dusty Circuit Lakewood OH 44107

10 2013-07-25 00:00:00.0 5648 PENDING\_PAYMENT 5648 Joshua Smith XXXXXXXXX XXXXXXXXX 864 Iron Spring Stead Memphis TN 38111

Time taken: 22.172 seconds, Fetched: 10 row(s)

select o.\* ,c.\* from orders o left outer join customers c on o.order\_customer\_id = c.customer\_id where o.order\_customer\_id is NULL;

Query ID = saurabhbhanwala\_20200903011653\_2ee54d00-7682-417d-86de-be105a3af8c5

Total jobs = 1

Execution log at: /tmp/saurabhbhanwala/saurabhbhanwala\_20200903011653\_2ee54d00-7682-417d-86de-be105a3af8c5.log

2020-09-03 01:17:05 Starting to launch local task to process map join; maximum memory = 1046478848

2020-09-03 01:17:06 Dump the side-table for tag: 1 with group count: 0 into file: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-16-53\_874\_4574363213529193316-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile31--.hashtable

2020-09-03 01:17:07 Uploaded 1 File to: file:/tmp/saurabhbhanwala/96b46327-8228-47dc-9e3b-657d7915b5f4/hive\_2020-09-03\_01-16-53\_874\_4574363213529193316-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile31--.hashtable (260 bytes)

2020-09-03 01:17:07 End of local task; Time Taken: 1.788 sec.

Execution completed successfully

MapredLocal task succeeded

Launching Job 1 out of 1

Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job\_1589064448439\_41762, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41762/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41762

Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0

2020-09-03 01:17:16,254 Stage-3 map = 0%, reduce = 0%

2020-09-03 01:17:21,433 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 3.03 sec

MapReduce Total cumulative CPU time: 3 seconds 30 msec

Ended Job = job\_1589064448439\_41762

MapReduce Jobs Launched:

Stage-Stage-3: Map: 1 Cumulative CPU: 3.03 sec HDFS Read: 3010806 HDFS Write: 0 SUCCESS

Total MapReduce CPU Time Spent: 3 seconds 30 msec

OK

o.order\_id o.order\_date o.order\_customer\_id o.order\_status c.customer\_id c.customer\_fname c.customer\_lname c.customer\_email c.customer\_password c.customer\_street c.customer\_city c.customer\_state c.customer\_zipcode

Time taken: 29.816 seconds

Analytics functions and windowing functions provide powerful query mechanisms.

* Typically uses over clause along with partition by or order by or both
* Aggregations can be performed
* Ranking per partition can be easily achieved
* Moving averages can be implemented with windowing functions

**What is Buckets?**

Buckets in hive is used in segregating of hive table-data into multiple files or directories. it is used for efficient querying.

* The data i.e. present in that partitions can be divided further into Buckets
* The division is performed based on Hash of particular columns that we selected in the table.
* Buckets use some form of Hashing algorithm at back end to read each record and place it into buckets
* In Hive, we have to enable buckets by using the **set.hive.enforce.bucketing=true;**

## What is Partitions?

Hive Partitions is a way to organizes tables into partitions by dividing tables into different parts based on partition keys.

Partition is helpful when the table has one or more Partition keys. Partition keys are basic elements for determining how the data is stored in the table.

**For Example**: -

"Client having Some E –commerce data which belongs to India operations in which each state (38 states) operations mentioned in as a whole. If we take state column as partition key and perform partitions on that India data as a whole, we can able to get Number of partitions (38 partitions) which is equal to number of states (38) present in India. Such that each state data can be viewed separately in partitions tables.

Sample Code Snippet for partitions

<http://hadooptutorial.info/hive-aggregate-functions/>

|  |
| --- |
| 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 |
|  | 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, pct\_revenue; |

Group by

|  |
| --- |
| 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; |

> 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;

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

> distribute by o.order\_date sort by o.order\_date, order\_revenue desc;

select 1, "Hello"

> union all

> select 2, "World"

> union all

> select 3, "Hello"

> union all

> select 4, "World";

<https://acadgild.com/blog/windowing-functions-in-hive>

select order\_item\_order\_id, order\_item\_product\_price, lag(order\_item\_product\_price, 1) over (partition by order\_item\_order\_id) as previous from order\_Items limit 10;

Query ID = saurabhbhanwala\_20200903065729\_628a681a-52bb-4f04-92a4-ac762c922848

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41835, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41835/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41835

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-03 06:57:38,595 Stage-1 map = 0%, reduce = 0%

2020-09-03 06:57:44,879 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.78 sec

2020-09-03 06:57:51,088 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 9.17 sec

MapReduce Total cumulative CPU time: 9 seconds 170 msec

Ended Job = job\_1589064448439\_41835

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 9.17 sec HDFS Read: 5419583 HDFS Write: 131 SUCCESS

Total MapReduce CPU Time Spent: 9 seconds 170 msec

OK

order\_item\_order\_id order\_item\_product\_price previous

1 299.98 NULL

2 50.0 NULL

2 129.99 50.0

2 199.99 129.99

4 49.98 NULL

4 59.99 49.98

4 24.99 59.99

4 50.0 24.99

5 299.98 NULL

5 59.99 299.98

select order\_item\_order\_id, order\_item\_product\_price, lead(order\_item\_product\_price, 1) over (partition by order\_item\_order\_id) as previous from order\_Items limit 10;

Query ID = saurabhbhanwala\_20200903065838\_72bd3588-6029-4c76-9fda-6a1bbd1fbdab

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_41837, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_41837/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_41837

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-03 06:58:46,205 Stage-1 map = 0%, reduce = 0%

2020-09-03 06:58:53,452 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.1 sec

2020-09-03 06:58:59,670 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.66 sec

MapReduce Total cumulative CPU time: 8 seconds 660 msec

Ended Job = job\_1589064448439\_41837

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.66 sec HDFS Read: 5419590 HDFS Write: 134 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 660 msec

OK

order\_item\_order\_id order\_item\_product\_price previous

1 299.98 NULL

2 50.0 129.99

2 129.99 199.99

2 199.99 NULL

4 49.98 59.99

4 59.99 24.99

4 24.99 50.0

4 50.0 NULL

5 299.98 59.99

5 59.99 49.98

# **FIRST\_VALUE**

Returns the expression value from the first row in the window. If your table has null values, you can use the IGNORE NULLS clause to return the first non-null value from the window. This same value is repeated for all result rows for the group. The return value is NULL if the input expression is NULL.

The PARTITION BY clause is optional. The ORDER BY clause is required. The window clause is optional.

select order\_item\_order\_id , first\_value(order\_item\_product\_price) over (partition by order\_item\_order\_id order by order\_item\_product\_price) first\_price from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905023445\_136682dd-b239-4d13-955d-0156b2773912

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42190, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42190/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42190

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:34:54,559 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:35:00,786 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.75 sec

2020-09-05 02:35:07,289 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.9 sec

MapReduce Total cumulative CPU time: 6 seconds 900 msec

Ended Job = job\_1589064448439\_42190

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.9 sec HDFS Read: 5419830 HDFS Write: 245 SUCCESS

Total MapReduce CPU Time Spent: 6 seconds 900 msec

OK

order\_item\_order\_id first\_price

1 299.98

2 50.0

2 50.0

2 50.0

4 24.99

4 24.99

4 24.99

4 24.99

5 49.98

5 49.98

5 49.98

5 49.98

5 49.98

7 15.99

7 15.99

7 15.99

8 49.98

8 49.98

8 49.98

8 49.98

9 99.99

9 99.99

9 99.99

10 21.99

10 21.99

10 21.99

10 21.99

10 21.99

11 39.99

11 39.99

LAST VALUE:

select order\_item\_order\_id , last\_value(order\_item\_product\_price) over (partition by order\_item\_order\_id order by order\_item\_product\_price) first\_price from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905023649\_e5d100ac-830f-40f6-9042-ff012396d5a0

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42191, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42191/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42191

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:36:58,730 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:37:04,950 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.1 sec

2020-09-05 02:37:11,165 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.49 sec

MapReduce Total cumulative CPU time: 8 seconds 490 msec

Ended Job = job\_1589064448439\_42191

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.49 sec HDFS Read: 5419823 HDFS Write: 257 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 490 msec

OK

order\_item\_order\_id first\_price

1 299.98

2 50.0

2 129.99

2 199.99

4 24.99

4 49.98

4 50.0

4 59.99

5 49.98

5 59.99

5 129.99

5 299.98

5 299.98

7 15.99

7 199.99

7 299.98

8 49.98

8 50.0

8 59.99

8 59.99

9 99.99

9 199.99

9 199.99

10 21.99

10 49.98

10 129.99

10 199.99

10 199.99

11 39.99

11 49.98

select order\_item\_order\_id, min(order\_item\_product\_price) over (partition by order\_item\_order\_id) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905024525\_662086d6-f261-4f60-9ae3-2c2d91f78a01

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42192, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42192/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42192

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:45:34,785 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:45:41,000 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.05 sec

2020-09-05 02:45:48,240 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.5 sec

MapReduce Total cumulative CPU time: 8 seconds 500 msec

Ended Job = job\_1589064448439\_42192

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.5 sec HDFS Read: 5419413 HDFS Write: 245 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 500 msec

OK

order\_item\_order\_id min\_window\_0

1 299.98

2 50.0

2 50.0

2 50.0

4 24.99

4 24.99

4 24.99

4 24.99

5 49.98

5 49.98

5 49.98

5 49.98

5 49.98

7 15.99

7 15.99

7 15.99

8 49.98

8 49.98

8 49.98

8 49.98

9 99.99

9 99.99

9 99.99

10 21.99

10 21.99

10 21.99

10 21.99

10 21.99

11 39.99

select order\_item\_order\_id, max(order\_item\_product\_price) over (partition by order\_item\_order\_id) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905024631\_d9934da9-e6fe-4ced-8faf-5990508c14d2

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42193, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42193/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42193

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:46:40,277 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:46:46,640 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.08 sec

2020-09-05 02:46:52,844 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.57 sec

MapReduce Total cumulative CPU time: 8 seconds 570 msec

Ended Job = job\_1589064448439\_42193

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.57 sec HDFS Read: 5419413 HDFS Write: 267 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 570 msec

OK

order\_item\_order\_id max\_window\_0

1 299.98

2 199.99

2 199.99

2 199.99

4 59.99

4 59.99

4 59.99

4 59.99

5 299.98

5 299.98

5 299.98

5 299.98

5 299.98

7 299.98

7 299.98

7 299.98

8 59.99

8 59.99

8 59.99

8 59.99

9 199.99

9 199.99

9 199.99

10 199.99

10 199.99

10 199.99

10 199.99

10 199.99

11 99.99

11 99.99

select order\_item\_order\_id, avg(order\_item\_product\_price) over (partition by order\_item\_order\_id) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905024812\_849b8457-aedd-4442-8a6e-a46b31a2df2e

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42194, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42194/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42194

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:48:20,824 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:48:27,114 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.11 sec

2020-09-05 02:48:33,317 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.37 sec

MapReduce Total cumulative CPU time: 8 seconds 370 msec

Ended Job = job\_1589064448439\_42194

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.37 sec HDFS Read: 5419532 HDFS Write: 623 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 370 msec

OK

order\_item\_order\_id avg\_window\_0

1 299.9800109863281

2 126.66000366210938

2 126.66000366210938

2 126.66000366210938

4 46.24000024795532

4 46.24000024795532

4 46.24000024795532

4 46.24000024795532

5 167.9840057373047

5 167.9840057373047

5 167.9840057373047

5 167.9840057373047

5 167.9840057373047

7 171.98667208353677

7 171.98667208353677

7 171.98667208353677

8 54.99000072479248

8 54.99000072479248

8 54.99000072479248

8 54.99000072479248

9 166.65666961669922

9 166.65666961669922

9 166.65666961669922

10 120.38800315856933

10 120.38800315856933

10 120.38800315856933

10 120.38800315856933

10 120.38800315856933

11 59.986000061035156

11 59.986000061035156

#### 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.

The below query will rank the closing prices of the stock for each ticker. The same you can see in the below screenshot.

select order\_item\_order\_id, order\_item\_product\_price, rank() over (partition by order\_item\_order\_id order by order\_item\_product\_price) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905025058\_cc033588-61cb-4b2b-b41c-3665b0dac348

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42196, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42196/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42196

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:51:07,332 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:51:13,551 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.21 sec

2020-09-05 02:51:19,768 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.82 sec

MapReduce Total cumulative CPU time: 8 seconds 820 msec

Ended Job = job\_1589064448439\_42196

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.82 sec HDFS Read: 5419779 HDFS Write: 317 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 820 msec

OK

order\_item\_order\_id order\_item\_product\_price rank\_window\_0

1 299.98 1

2 50.0 1

2 129.99 2

2 199.99 3

4 24.99 1

4 49.98 2

4 50.0 3

4 59.99 4

5 49.98 1

5 59.99 2

5 129.99 3

5 299.98 4

5 299.98 4

7 15.99 1

7 199.99 2

7 299.98 3

8 49.98 1

8 50.0 2

8 59.99 3

8 59.99 3

9 99.99 1

9 199.99 2

9 199.99 2

10 21.99 1

10 49.98 2

10 129.99 3

10 199.99 4

10 199.99 4

11 39.99 1

11 49.98 2

#### Row\_number

Row number will return the continuous sequence of numbers for all the rows of the result set of the over clause.

From the below query, you will get the ticker, closing price and its row number for each ticker.

select order\_item\_order\_id, order\_item\_product\_price, row\_number() over (partition by order\_item\_order\_id order by order\_item\_product\_price) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905025321\_cb676124-247e-457d-88d5-9ec21874e55a

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42197, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42197/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42197

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 02:53:30,536 Stage-1 map = 0%, reduce = 0%

2020-09-05 02:53:36,830 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.22 sec

2020-09-05 02:53:43,036 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.71 sec

MapReduce Total cumulative CPU time: 8 seconds 710 msec

Ended Job = job\_1589064448439\_42197

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.71 sec HDFS Read: 5419648 HDFS Write: 317 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 710 msec

OK

order\_item\_order\_id order\_item\_product\_price row\_number\_window\_0

1 299.98 1

2 50.0 1

2 129.99 2

2 199.99 3

4 24.99 1

4 49.98 2

4 50.0 3

4 59.99 4

5 49.98 1

5 59.99 2

5 129.99 3

5 299.98 4

5 299.98 5

7 15.99 1

7 199.99 2

7 299.98 3

8 49.98 1

8 50.0 2

8 59.99 3

8 59.99 4

9 99.99 1

9 199.99 2

9 199.99 3

10 21.99 1

10 49.98 2

10 129.99 3

10 199.99 4

10 199.99 5

11 39.99 1

11 49.98 2

#### 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. The same you can see in the below screenshot.

select order\_item\_order\_id, order\_item\_product\_price, dense\_rank() over (partition by order\_item\_order\_id order by order\_item\_product\_price) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905030149\_40b4d085-4f45-421c-b453-c7f338b7aeac

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42198, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42198/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42198

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 03:01:58,326 Stage-1 map = 0%, reduce = 0%

2020-09-05 03:02:04,546 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.99 sec

2020-09-05 03:02:10,747 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.58 sec

MapReduce Total cumulative CPU time: 8 seconds 580 msec

Ended Job = job\_1589064448439\_42198

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.58 sec HDFS Read: 5419819 HDFS Write: 317 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 580 msec

OK

order\_item\_order\_id order\_item\_product\_price dense\_rank\_window\_0

1 299.98 1

2 50.0 1

2 129.99 2

2 199.99 3

4 24.99 1

4 49.98 2

4 50.0 3

4 59.99 4

5 49.98 1

5 59.99 2

5 129.99 3

5 299.98 4

5 299.98 4

7 15.99 1

7 199.99 2

7 299.98 3

8 49.98 1

8 50.0 2

8 59.99 3

8 59.99 3

9 99.99 1

9 199.99 2

9 199.99 2

10 21.99 1

10 49.98 2

10 129.99 3

10 199.99 4

10 199.99 4

11 39.99 1

11 49.98 2

#### Cume\_dist

It returns the cumulative distribution of a value. It results from 0 to 1.

select order\_item\_order\_id, order\_item\_product\_price, cume\_dist() over (partition by order\_item\_order\_id order by order\_item\_product\_price) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905030427\_34ea6e80-3678-4169-9789-6163847261f0

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42199, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42199/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42199

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 03:04:36,479 Stage-1 map = 0%, reduce = 0%

2020-09-05 03:04:42,690 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.21 sec

2020-09-05 03:04:48,894 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.96 sec

MapReduce Total cumulative CPU time: 8 seconds 960 msec

Ended Job = job\_1589064448439\_42199

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.96 sec HDFS Read: 5419916 HDFS Write: 455 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 960 msec

OK

order\_item\_order\_id order\_item\_product\_price cume\_dist\_window\_0

1 299.98 1.0

2 50.0 0.3333333333333333

2 129.99 0.6666666666666666

2 199.99 1.0

4 24.99 0.25

4 49.98 0.5

4 50.0 0.75

4 59.99 1.0

5 49.98 0.2

5 59.99 0.4

5 129.99 0.6

5 299.98 1.0

5 299.98 1.0

7 15.99 0.3333333333333333

7 199.99 0.6666666666666666

7 299.98 1.0

8 49.98 0.25

8 50.0 0.5

8 59.99 1.0

8 59.99 1.0

9 99.99 0.3333333333333333

9 199.99 1.0

9 199.99 1.0

10 21.99 0.2

10 49.98 0.4

10 129.99 0.6

10 199.99 1.0

10 199.99 1.0

11 39.99 0.2

11 49.98 0.6

#### 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 order\_item\_order\_id, order\_item\_product\_price, percent\_rank() over (partition by order\_item\_order\_id order by order\_item\_product\_price) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905030745\_1a14dba1-448b-49c1-91bb-f2c123e9197d

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42200, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42200/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42200

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 03:07:53,252 Stage-1 map = 0%, reduce = 0%

2020-09-05 03:08:00,497 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.88 sec

2020-09-05 03:08:06,703 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.35 sec

MapReduce Total cumulative CPU time: 7 seconds 350 msec

Ended Job = job\_1589064448439\_42200

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.35 sec HDFS Read: 5419937 HDFS Write: 459 SUCCESS

Total MapReduce CPU Time Spent: 7 seconds 350 msec

OK

order\_item\_order\_id order\_item\_product\_price percent\_rank\_window\_0

1 299.98 0.0

2 50.0 0.0

2 129.99 0.5

2 199.99 1.0

4 24.99 0.0

4 49.98 0.3333333333333333

4 50.0 0.6666666666666666

4 59.99 1.0

5 49.98 0.0

5 59.99 0.25

5 129.99 0.5

5 299.98 0.75

5 299.98 0.75

7 15.99 0.0

7 199.99 0.5

7 299.98 1.0

8 49.98 0.0

8 50.0 0.3333333333333333

8 59.99 0.6666666666666666

8 59.99 0.6666666666666666

9 99.99 0.0

9 199.99 0.5

9 199.99 0.5

10 21.99 0.0

10 49.98 0.25

10 129.99 0.5

10 199.99 0.75

10 199.99 0.75

11 39.99 0.0

11 49.98 0.25

#### 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 order\_item\_order\_id, order\_item\_product\_price, ntile(5) over (partition by order\_item\_order\_id order by order\_item\_product\_price) from order\_items limit 30;

Query ID = saurabhbhanwala\_20200905031115\_606d7d6b-f415-455e-b4e6-de4ef4bc0cc9

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapreduce.job.reduces=<number>

Starting Job = job\_1589064448439\_42201, Tracking URL = http://rm01.itversity.com:19088/proxy/application\_1589064448439\_42201/

Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job -kill job\_1589064448439\_42201

Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1

2020-09-05 03:11:24,882 Stage-1 map = 0%, reduce = 0%

2020-09-05 03:11:31,088 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.0 sec

2020-09-05 03:11:37,290 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.7 sec

MapReduce Total cumulative CPU time: 8 seconds 700 msec

Ended Job = job\_1589064448439\_42201

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.7 sec HDFS Read: 5419686 HDFS Write: 317 SUCCESS

Total MapReduce CPU Time Spent: 8 seconds 700 msec

OK

order\_item\_order\_id order\_item\_product\_price ntile\_window\_0

1 299.98 1

2 50.0 1

2 129.99 2

2 199.99 3

4 24.99 1

4 49.98 2

4 50.0 3

4 59.99 4

5 49.98 1

5 59.99 2

5 129.99 3

5 299.98 4

5 299.98 5

7 15.99 1

7 199.99 2

7 299.98 3

8 49.98 1

8 50.0 2

8 59.99 3

8 59.99 4

9 99.99 1

9 199.99 2

9 199.99 3

10 21.99 1

10 49.98 2

10 129.99 3

10 199.99 4

10 199.99 5

11 39.99 1

11 49.98 2

Here are the steps to create data frame and run queries against it

* Read data from HDFS
* Apply map transformation to convert each record into the tuple
* Use toDF function to convert RDD into Data Frame
* Register Data Frame as a temp table
* Run queries against temp table using sqlContext.sql

**Objectives**

* Create Spark DataFrames from an existing RDD
* Perform operations on a DataFrame
* Write a Spark SQL application
* Use Hive with ORC from Spark SQL
* Write a Spark SQL application that reads and writes data from Hive  
  tables

pyspark --master yarn --conf spark.ui.port=12576 --executor-memory 2G --num-executors 2

import os

>>> os.system("ls -all /data/retail\_db/")

total 32

drwxr-xr-x 8 root root 4096 Feb 20 2017 .

drwxr-xr-x 22 root root 4096 Aug 16 00:05 ..

drwxr-xr-x 2 root root 4096 Feb 20 2017 categories

drwxr-xr-x 2 root root 4096 Feb 20 2017 customers

drwxr-xr-x 2 root root 4096 Feb 20 2017 departments

drwxr-xr-x 2 root root 4096 Feb 20 2017 order\_items

drwxr-xr-x 2 root root 4096 Feb 20 2017 orders

drwxr-xr-x 2 root root 4096 Feb 20 2017 products

from pyspark.sql import Row

>>> ordersRDD = sc.textFile("retail\_db/orders")

>>> for i in ordersRDD.take(10): print(i)

...

1,2013-07-25 00:00:00.0,11599,CLOSED

2,2013-07-25 00:00:00.0,256,PENDING\_PAYMENT

3,2013-07-25 00:00:00.0,12111,COMPLETE

4,2013-07-25 00:00:00.0,8827,CLOSED

5,2013-07-25 00:00:00.0,11318,COMPLETE

6,2013-07-25 00:00:00.0,7130,COMPLETE

7,2013-07-25 00:00:00.0,4530,COMPLETE

8,2013-07-25 00:00:00.0,2911,PROCESSING

9,2013-07-25 00:00:00.0,5657,PENDING\_PAYMENT

10,2013-07-25 00:00:00.0,5648,PENDING\_PAYMENT

ordersDF = ordersRDD.map(lambda p: Row(order\_id = int(p.split(",")[0]), order\_date= p.split(",")[1], order\_customer\_id =int(p.split(",")[2]), order\_status =p.split(",")[3])).toDF()

>>> type(ordersDF)

<class 'pyspark.sql.dataframe.DataFrame'>

ordersDF = ordersRDD.map(lambda p: Row(order\_id = int(p.split(",")[0]), order\_date= p.split(",")[1], order\_customer\_id =int(p.split(",")[2]), order\_status =p.split(",")[3])).toDF()

>>> type(ordersDF)

<class 'pyspark.sql.dataframe.DataFrame'>

>>> ordersDF.registerTempTable("ordersDF\_table")

>>> sqlContext.sql("select \* from ordersDF\_table limit 10").show()

+-----------------+--------------------+--------+---------------+

|order\_customer\_id| order\_date|order\_id| order\_status|

+-----------------+--------------------+--------+---------------+

| 11599|2013-07-25 00:00:...| 1| CLOSED|

| 256|2013-07-25 00:00:...| 2|PENDING\_PAYMENT|

| 12111|2013-07-25 00:00:...| 3| COMPLETE|

| 8827|2013-07-25 00:00:...| 4| CLOSED|

| 11318|2013-07-25 00:00:...| 5| COMPLETE|

| 7130|2013-07-25 00:00:...| 6| COMPLETE|

| 4530|2013-07-25 00:00:...| 7| COMPLETE|

| 2911|2013-07-25 00:00:...| 8| PROCESSING|

| 5657|2013-07-25 00:00:...| 9|PENDING\_PAYMENT|

| 5648|2013-07-25 00:00:...| 10|PENDING\_PAYMENT|

+-----------------+--------------------+--------+---------------+

productsRaw = open("/data/retail\_db/products/part-00000").read().splitlines()

productsRaw[:10]

['1,2,Quest Q64 10 FT. x 10 FT. Slant Leg Instant U,,59.98,http://images.acmesports.sports/Quest+Q64+10+FT.+x+10+FT.+Slant+Leg+Instant+Up+Canopy', "2,2,Under Armour Men's Highlight MC Football Clea,,129.99,http://images.acmesports.sports/Under+Armour+Men%27s+Highlight+MC+Football+Cleat", "3,2,Under Armour Men's Renegade D Mid Football Cl,,89.99,http://images.acmesports.sports/Under+Armour+Men%27s+Renegade+D+Mid+Football+Cleat", "4,2,Under Armour Men's Renegade D Mid Football Cl,,89.99,http://images.acmesports.sports/Under+Armour+Men%27s+Renegade+D+Mid+Football+Cleat", '5,2,Riddell Youth Revolution Speed Custom Footbal,,199.99,http://images.acmesports.sports/Riddell+Youth+Revolution+Speed+Custom+Football+Helmet', "6,2,Jordan Men's VI Retro TD Football Cleat,,134.99,http://images.acmesports.sports/Jordan+Men%27s+VI+Retro+TD+Football+Cleat", '7,2,Schutt Youth Recruit Hybrid Custom Football H,,99.99,http://images.acmesports.sports/Schutt+Youth+Recruit+Hybrid+Custom+Football+Helmet+2014', "8,2,Nike Men's Vapor Carbon Elite TD Football Cle,,129.99,http://images.acmesports.sports/Nike+Men%27s+Vapor+Carbon+Elite+TD+Football+Cleat", '9,2,Nike Adult Vapor Jet 3.0 Receiver Gloves,,50.0,http://images.acmesports.sports/Nike+Adult+Vapor+Jet+3.0+Receiver+Gloves', "10,2,Under Armour Men's Highlight MC Football Clea,,129.99,http://images.acmesports.sports/Under+Armour+Men%27s+Highlight+MC+Football+Cleat"]

>>> productsRDD = sc.parallelize(productsRaw)

>>> for i in productsRDD.take(10): print(i)

...

1,2,Quest Q64 10 FT. x 10 FT. Slant Leg Instant U,,59.98,http://images.acmesports.sports/Quest+Q64+10+FT.+x+10+FT.+Slant+Leg+Instant+Up+Canopy

2,2,Under Armour Men's Highlight MC Football Clea,,129.99,http://images.acmesports.sports/Under+Armour+Men%27s+Highlight+MC+Football+Cleat

3,2,Under Armour Men's Renegade D Mid Football Cl,,89.99,http://images.acmesports.sports/Under+Armour+Men%27s+Renegade+D+Mid+Football+Cleat

4,2,Under Armour Men's Renegade D Mid Football Cl,,89.99,http://images.acmesports.sports/Under+Armour+Men%27s+Renegade+D+Mid+Football+Cleat

5,2,Riddell Youth Revolution Speed Custom Footbal,,199.99,http://images.acmesports.sports/Riddell+Youth+Revolution+Speed+Custom+Football+Helmet

6,2,Jordan Men's VI Retro TD Football Cleat,,134.99,http://images.acmesports.sports/Jordan+Men%27s+VI+Retro+TD+Football+Cleat

7,2,Schutt Youth Recruit Hybrid Custom Football H,,99.99,http://images.acmesports.sports/Schutt+Youth+Recruit+Hybrid+Custom+Football+Helmet+2014

8,2,Nike Men's Vapor Carbon Elite TD Football Cle,,129.99,http://images.acmesports.sports/Nike+Men%27s+Vapor+Carbon+Elite+TD+Football+Cleat

9,2,Nike Adult Vapor Jet 3.0 Receiver Gloves,,50.0,http://images.acmesports.sports/Nike+Adult+Vapor+Jet+3.0+Receiver+Gloves

10,2,Under Armour Men's Highlight MC Football Clea,,129.99,http://images.acmesports.sports/Under+Armour+Men%27s+Highlight+MC+Football+Cleat

>>> productsDF = productsRDD.map(lambda p: Row(product\_id=int(p.split(",")[0]), productsName=p.split(",")[2])).toDF()

>>> productsDF.registerTempTable("productsDF\_table")

sqlContext.sql("select productsName, count(1) from productsDF\_table group by productsName limit 20").show()

+--------------------+---+

| productsName|\_c1|

+--------------------+---+

|Top Flite Women's...| 1|

|Nike Men's Pittsb...| 1|

|Nike Women's Pro ...| 2|

|Top Flite Women's...| 1|

|adidas Men's Germ...| 8|

|Riddell Youth Rev...| 1|

|"Nike Men's New Y...| 1|

|Nike Men's Home G...| 2|

|Nike Men's Mercur...| 2|

|Lady Hagen Women'...| 1|

|Glove It Women's ...| 1|

|Quest 12' x 12' D...| 8|

|Pelican Trailblaz...| 3|

|Toronto FC Team C...| 1|

|Under Armour Men'...| 1|

|Cleveland Golf Wo...| 1|

|Easton S1 Youth B...| 1|

|Cobra Women's AMP...| 1|

|Nike Men's Free T...| 1|

|Fitness Gear 7' O...| 1|

+--------------------+---+

sqlContext is already connected to hiveContext

sqlContext.sql("show tables").show()

+--------------------+-----------+

| tableName|isTemporary|

+--------------------+-----------+

| ordersdf\_table| true|

| productsdf\_table| true|

| 0001\_table\_name| false|

| 001\_order\_ga| false|

| 00\_order\_ga| false|

| 3region| false|

| a| false|

| a\_order| false|

| aa\_test| false|

| aabidajanee\_names| false|

|aabidajanee\_names...| false|

|aabidajanee\_names...| false|

| aabidajanee\_people| false|

|aabidajanee\_stock...| false|

|aaronolivares12\_n...| false|

|aaronolivares12\_n...| false|

|aaronolivares12\_n...| false|

|aaronolivares12\_s...| false|

| abb\_traversal\_input| false|

| abc| false|

+--------------------+-----------+

only showing top 20 rows

sqlContext.sql("use saurabhbhanwala\_retail\_db\_text")

DataFrame[result: string]

>>> sqlContext.sql("show tables").show()

+----------------+-----------+

| tableName|isTemporary|

+----------------+-----------+

| ordersdf\_table| true|

|productsdf\_table| true|

| customers| false|

| order\_items| false|

| orders| false|

+----------------+-----------+

**Spark SQL application – Hive Context**

* Get daily revenue by product considering completed and closed  
  orders.
  + PRODUCTS have to be read from the local file system. DataFrame need to  
    be created.
  + Join ORDERS, ORDER\_ITEMS
  + Filter on ORDER\_STATUS
* Data need to be sorted in ascending order by date and then  
  descending order by revenue computed for each product for each day.
  + Sort data by order\_date in ascending order and then daily revenue per  
    product in descending order
* Once tables are registered, we can
  + Run all standard queries
  + Join temp tables with other hive tables

sqlContext.sql("select o.order\_date, p.productsName, sum(oi.order\_item\_subtotal) as daily\_revenue\_per\_product \

from order\_items oi join productsdf\_table p on oi.order\_item\_product\_id = p.product\_id join orders o on o.order\_id = oi.order\_item\_order\_id \

where o.order\_status in ('COMPLETE','CLOSED') \

group by o.order\_date, p.productsName order by o.order\_date, daily\_revenue\_per\_product desc") .show()

dailyRevenueDF = sqlContext.sql("select o.order\_date, p.productsName, sum(oi.order\_item\_subtotal) as daily\_revenue\_per\_product \

... from order\_items oi join productsdf\_table p on oi.order\_item\_product\_id = p.product\_id join orders o on o.order\_id = oi.order\_item\_order\_id \

... where o.order\_status in ('COMPLETE','CLOSED') \

... group by o.order\_date, p.productsName order by o.order\_date, daily\_revenue\_per\_product desc")

dailyRevenueDF.insertInto("saurbahbhanwala\_daily\_revenue\_per\_products.daily\_revenue")