**--Create Database**

CREATE DATABASE IF NOT EXISTS hive\_db\_demo

COMMENT 'weclouddata hive database'

WITH DBPROPERTIES ('creator'='weclouddata','date'='2017-03-03');

**--Describe and Use Database**

SHOW DATABASES;

DESCRIBE DATABASE hive\_db\_demo;

USE hive\_db\_demo;

**--Drop the Database**

DROP DATABASE IF EXISTS hive\_db\_demo;

**--Create Customer Table**

CREATE TABLE IF NOT EXISTS customer(

id string,

credits ARRAY<string>,

profile STRUCT<name:string,gender:string,dob:date,height:int,citizenship:string,student:boolean>,

contact MAP<string,string>,

address STRUCT<no:string,street:string,city:string,province:string,country:string,postal:string>,

preferpayment string

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '$'

COLLECTION ITEMS TERMINATED BY '^'

MAP KEYS TERMINATED BY '#';

**--Prepare Data File and Save into /root/customer.txt**

jiangb$visa^master^debit^cash$bin jiang^male^2010-01-01^170^canada^true$cell#4161111010^home#9051011111^office#4168888888$100^highway 7^markham^on^canada^l155j1$visa

michael$master^debit^cash$michael liu^male^1990-05-05^180^canada^false$cell#6471111888^home#4165011111$888^sheppard ave^scarborough^on^canada^m1l5k1$cash

tina$visa^master^debit$tina fan^female^2000-01-09^160^china^false$cell#4161871010$100^16th ave^richmond hill^on^canada^k155j1$debt

ryan$cash$ryan witcom^male^1980-09-05^185^france^false$home#9056011111$3000^victoria park ave^north york^on^canada^g1l5l1$master

**--Load Data into Customer Table**

LOAD DATA LOCAL INPATH '/root/customer.txt' OVERWRITE INTO TABLE customer;

**--Query the Customer table**

SELECT \* FROM customer;

**--Query the ARRAY in the Customer table**

SELECT credits FROM customer;

SELECT credits[0] AS credit0, credits[1] AS credit1, credits[2] AS credit2, credits[3] AS credit3 FROM customer;

**--Show Column Name**

set hive.cli.print.header=true;

**--Query the STRUCT in the Customer table**

SELECT profile FROM customer;

SELECT profile.name, profile.gender, profile.dob, profile.height, profile.citizenship, profile.student FROM customer;

SELECT address FROM customer;

SELECT id, address.no, address.street, address.city, address.province, address.country, address.postal FROM customer;

**--Query the MAP in the Customer table**

SELECT contact FROM customer;

SELECT id, contact['cell'] AS cell\_phone, contact['home'] AS home\_phone, contact['office'] AS office\_phone FROM customer;

**--Create Orders Table**

CREATE TABLE IF NOT EXISTS orders(

id string,

userid string,

vender string,

amount double,

quantity int

)

PARTITIONED BY (city string, purchasedate string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**--Insert Data into Orders table**

INSERT INTO orders partition(city, purchasedate) VALUES ('00001', 'jiangb', 'walmart', 100.00, 5, 'toronto', '20160501');

INSERT INTO orders partition(city, purchasedate) VALUES ('00002', 'jiangb', 'loblaws', 300.00, 15, 'toronto', '20170303');

INSERT INTO orders partition(city, purchasedate) VALUES ('00003', 'michael', 'apple', 55.10, 3, 'paris', '20150607');

INSERT INTO orders partition(city, purchasedate) VALUES ('00004', 'tina', 'walmart', 6000.00, 55, 'toronto', '20160501');

INSERT INTO orders partition(city, purchasedate) VALUES ('00005', 'ryan', 'loblaws', 700.00, 7, 'toronto', '20170203');

INSERT INTO orders partition(city, purchasedate) VALUES ('00006', 'tina', 'apple', 155.10, 1, 'paris', '20170707');

INSERT INTO orders partition(city, purchasedate) VALUES ('00007', 'jiangb', 'walmart', 100.00, 2, 'toronto', '20160501');

INSERT INTO orders partition(city, purchasedate) VALUES ('00008', 'jiangb', 'loblaws', 990.00, 12, 'toronto', '20170303');

INSERT INTO orders partition(city, purchasedate) VALUES ('00009', 'michael', 'apple', 555.10, 9, 'paris', '20150607');

INSERT INTO orders partition(city, purchasedate) VALUES ('000010', 'tina', 'walmart', 80.00, 6, 'toronto', '20160501');

INSERT INTO orders partition(city, purchasedate) VALUES ('000011', 'ryan', 'loblaws', 7.00, 1, 'toronto', '20170203');

INSERT INTO orders partition(city, purchasedate) VALUES ('000012', 'tina', 'apple', 378.10, 8, 'paris', '20170707');

**--Query the Orders table**

SELECT \* FROM orders;

**--Show Orders table partitions**

SHOW PARTITIONS orders;

**--Add partitions to Orders**

ALTER TABLE orders ADD

PARTITION (city='montreal', purchasedate='20000101')

PARTITION (city='montreal', purchasedate='20010909');

**--Drop partitions from Orders**

ALTER TABLE orders DROP PARTITION (city='montreal', purchasedate='20000101');

ALTER TABLE orders DROP PARTITION (city='montreal', purchasedate='20010909');

**--Prepare Data File and save into /root/orders.txt**

000013,tina,walmart,765,6

000014,tina,walmart,765,6

000015,tina,walmart,765,6

**--Upload into HDFS /tmp/orders.txt**

hadoop fs -copyFromLocal /root/orders.txt /tmp/orders.txt

hadoop fs -chmod 777 /tmp/orders.txt

**--Load data to the partition**

LOAD DATA INPATH '/tmp/orders.txt' INTO TABLE orders PARTITION (city='montreal', purchasedate='20000101');

**--Create a Orders History table with bucketing**

CREATE TABLE IF NOT EXISTS orders\_history(

id string,

userid string,

vender string,

amount double,

quantity int

)

PARTITIONED BY (city string, purchasedate string)

CLUSTERED BY (userid) INTO 2 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

SET hive.enforce.bucketing = true;

INSERT OVERWRITE TABLE orders\_history partition(city, purchasedate) SELECT \* FROM orders;

------这里出错了——

**--Dynamic partition is not enabled by default**

SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode=nostrict;

**--Prepare Data File and save into /root/customer\_action.txt**

jiangb,jsessionid000000000001,purchase,2017-03-03 00:17:13,chrome,google

jiangb,jsessionid000000000001,product,2017-03-03 00:17:33,chrome,google

michael,jsessionid000000000003,product,2017-03-03 00:18:33,ie,yahoo

tina,jsessionid000000000005,home,2017-03-03 00:19:33,firefox,facebook

ryan,jsessionid000000000006,info,2017-03-03 00:19:50,safari,linkedin

jiangb,jsessionid000000000011,shoppingcart,2017-03-03 00:19:33,chrome,google

**--Upload into HDFS /tmp/customeraction/edate=20170303**

hadoop fs -mkdir /tmp/customeraction

hadoop fs -mkdir /tmp/customeraction/edate=20170303

hadoop fs -copyFromLocal /root/customer\_action.txt /tmp/customeraction/edate=20170303/customer\_action.txt

hadoop fs -chmod -R 777 /tmp/customeraction

**--Create Customer Action external table and load the data**

CREATE EXTERNAL TABLE IF NOT EXISTS customer\_action(

userid string,

sessionid string,

page string,

visitedtime timestamp,

browser string,

referer string

)

partitioned by(edate string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/tmp/customeraction';

**--Refresh Custom Action table with new partition**

MSCK REPAIR TABLE customer\_action;

**--Prepare Data File and save into /root/customer\_action.txt**

jiangb,jsessionid100000000001,purchase,2017-03-02 00:17:13,chrome,google

jiangb,jsessionid100000000001,product,2017-03-02 00:17:33,chrome,google

michael,jsessionid100000000003,product,2017-03-02 00:18:33,ie,yahoo

tina,jsessionid100000000005,home,2017-03-02 00:19:33,firefox,facebook

ryan,jsessionid100000000006,info,2017-03-02 00:19:50,safari,linkedin

jiangb,jsessionid100000000011,shoppingcart,2017-03-02 00:19:33,chrome,google

**--Upload into HDFS /tmp/customeraction/edate=20170302**

hadoop fs -mkdir /tmp/customeraction/edate=20170302

hadoop fs -copyFromLocal /root/customer\_action.txt /tmp/customeraction/edate=20170302/customer\_action.txt

**--Refresh Custom Action table with new partition**

MSCK REPAIR TABLE customer\_action;

**--JOIN between two tables**

SELECT customer.id as userid, customer.profile.name as customer\_name, orders.id as orderid FROM customer JOIN orders ON customer.id = orders.userid;

SELECT customer.id as userid, customer.profile.name as customer\_name, orders.id as orderid FROM customer JOIN orders ON customer.id = orders.userid sort by orderid;

SELECT customer.id as userid, customer.profile.name as customer\_name, orders.id as orderid FROM customer JOIN orders ON customer.id = orders.userid order by orderid;

SELECT customer.id as userid, customer.profile.name as customer\_name, count(orders.id) as order\_number, sum(orders.amount) as total\_amount FROM customer JOIN orders ON customer.id = orders.userid group by customer.id, customer.profile.name;

SELECT customer.id as userid, customer.profile.name as customer\_name, collect\_set(orders.id), count(orders.id) as order\_number, sum(orders.amount) as total\_amount FROM customer JOIN orders ON customer.id = orders.userid group by customer.id, customer.profile.name;

**--MAP JOIN enabled by query hint**

SELECT /\*+ MAPJOIN(customer)\*/ customer.id, customer\_action.\* FROM customer JOIN customer\_action WHERE customer.id = customer\_action.userid;

**--Create Table With LIKE**

CREATE TABLE customer\_like LIKE customer;

DESCRIBE customer\_like;

INSERT INTO customer\_like

SELECT 'peter' as id, array('visa') as credits, named\_struct('name','peter sampas','gender','male','dob',CAST('1995-01-09' as date),'height',190,'citizenship','usa','student',false) as profile,

map('cell','4161506788') as contact,

named\_struct('no','999','street','eglinton ave','city','toronto','province','on','country','canada','postal','m1p5l1') as address, 'visa' as preferpayment

FROM customer limit 1;

**--Create Table With CTAS**

CREATE TABLE customer\_ctas AS SELECT \* FROM customer;

**--Create Table with Common Table Expression (CTE)**

CREATE TABLE customer\_cte AS

WITH male AS (SELECT profile.name as name FROM customer WHERE profile.gender= 'male' and id = 'jiangb'),

female AS (SELECT profile.name as name FROM customer WHERE profile.gender= 'female')

SELECT \* FROM male UNION ALL select \* FROM female;

**--Alter Table File Format to Parquet**

CREATE TABLE IF NOT EXISTS orders\_parquet(

id string,

userid string,

vender string,

amount double,

quantity int

)

PARTITIONED BY (city string, purchasedate string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS PARQUET;

INSERT INTO orders\_parquet partition(city,purchasedate) select \* from orders;

**--Create Hive view**

CREATE VIEW orders\_view AS select id, userid from orders;

--Nest SELECT

SELECT \* FROM (

SELECT \* FROM customer

WHERE profile.gender = 'male'

) male;

**--Subquery**

SELECT \* FROM orders WHERE orders.userid IN (SELECT id FROM customer WHERE customer.profile.gender = 'female');

**--UNION**

SELECT customer.id as username FROM customer UNION ALL SELECT orders.userid as username FROM orders;

SELECT DISTINCT a.username FROM (SELECT customer.id as username FROM customer UNION ALL SELECT orders.userid as username FROM orders) a;

**--INTERCEPT**

SELECT customer.id as username FROM customer JOIN orders ON customer.id = orders.userid;

**--MINUS**

SELECT customer.id as username FROM customer LEFT JOIN orders ON customer.id = orders.userid WHERE orders.userid IS NULL;

SELECT customer.id as username FROM customer LEFT JOIN orders ON customer.id = orders.userid WHERE orders.userid IS NULL

UNION ALL

SELECT orders.userid as username FROM customer RIGHT JOIN orders ON customer.id = orders.userid WHERE customer.id IS NULL;

**--INSERTS By Scanning the table only once**

FROM customer

INSERT OVERWRITE TABLE customer\_ctas

SELECT \* WHERE profile.gender = 'male'

INSERT OVERWRITE TABLE customer\_like

SELECT \* WHERE profile.gender = 'female';

**--Insert to local files with default row separators**

INSERT OVERWRITE LOCAL DIRECTORY '/root/customer\_output' SELECT \* FROM customer;

FROM customer

INSERT OVERWRITE LOCAL DIRECTORY '/root/customer\_output1'

SELECT \*

INSERT OVERWRITE LOCAL DIRECTORY '/root/customer\_output2'

SELECT \*;

**--Aggregation without GROUP BY columns**

SELECT count(\*) AS orders FROM orders;

**--HAVING**

SELECT userid, count(\*) as total\_order FROM orders GROUP BY userid HAVING count(\*) > 1;

**--GROUPING SETS**

SELECT city, purchasedate, userid, count(id) total\_order FROM orders

GROUP BY city, purchasedate, userid GROUPING SETS (city, purchasedate, userid);

**--ROLLUP**

SELECT city, purchasedate, userid, count(id) total\_order FROM orders

GROUP BY city, purchasedate, userid WITH ROLLUP;

**--CUBE**

SELECT city, purchasedate, userid, count(id) total\_order FROM orders

GROUP BY city, purchasedate, userid WITH CUBE;

**--Analytic Functions**

SELECT city, userid, id as orderid, amount,

COUNT(\*) OVER (PARTITION BY city) AS order\_bycity,

SUM(amount) OVER(PARTITION BY city ORDER BY city) AS amount\_bycity,

SUM(amount) OVER(ORDER BY city) AS total\_amount1,

SUM(amount) OVER(ORDER BY city, userid rows unbounded preceding) AS total\_amount2

FROM orders

ORDER BY city, userid;

SELECT city, userid, id as orderid, amount,

RANK() OVER (PARTITION BY city ORDER BY amount) AS rank,

DENSE\_RANK() OVER (PARTITION BY city ORDER BY amount) AS dense\_rank,

ROW\_NUMBER() OVER () AS row\_num,

ROUND((CUME\_DIST() OVER (PARTITION BY city ORDER BY amount)), 1) AS cume\_dist,

PERCENT\_RANK() OVER(PARTITION BY city ORDER BY amount) AS percent\_rank

FROM orders

ORDER BY city;

SELECT city, userid, id as orderid, amount,

LEAD(amount, 2) OVER(PARTITION BY city ORDER BY amount) AS lead,

LAG(amount, 2, 0) OVER(PARTITION BY city ORDER BY amount) AS lag,

FIRST\_VALUE(amount) OVER (PARTITION BY city ORDER BY amount) AS first\_value,

LAST\_VALUE(amount) OVER (PARTITION BY city ORDER BY amount) AS last\_value\_default,

LAST\_VALUE(amount) OVER (PARTITION BY city ORDER BY amount RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)

AS last\_value FROM orders ORDER BY city;

SELECT city, userid, id as orderid, amount,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) w1,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN 2 PRECEDING AND UNBOUNDED FOLLOWING) w2,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN 1 PRECEDING AND 2 FOLLOWING) w3,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) w4,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) w5,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN CURRENT ROW AND CURRENT ROW) w6,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING) w7,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) w8,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) w9,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN UNBOUNDED PRECEDING AND 1 FOLLOWING) w10,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) w11,

MIN(amount) OVER (PARTITION BY city ORDER BY userid ROWS 2 PRECEDING) w12

FROM orders ORDER BY city, userid;

**--EXPLAIN statement**

EXPLAIN SELECT purchasedate, count(\*) FROM orders WHERE city='toronto' GROUP BY purchasedate;

**--ANALYZE statement**

ANALYZE TABLE customer COMPUTE STATISTICS;

ANALYZE TABLE orders PARTITION(city='toronto', purchasedate='20160501') COMPUTE STATISTICS;

**--Data file optimization**

SET hive.exec.compress.output=true;

SET io.seqfile.compression.type=BLOCK;

**--Compression**

SET hive.exec.compress.intermediate=true;

SET hive.intermediate.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;

SET hive.exec.compress.output=true;

SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;

--Storage optimization

SET hive.exec.mode.local.auto=true;

SET hive.exec.mode.local.auto.inputbytes.max=50000000;

SET hive.exec.mode.local.auto.input.files.max=5;

**--JVM reuse**

SET mapred.job.reuse.jvm.num.tasks = 5;

**--Parallel running job**

SET hive.exec.parallel=true;

SET hive.exec.parallel.thread.number=16;

**--Map Join**

SET hive.auto.convert.join=true;

SET hive.mapjoin.smalltable.filesize=600000000;

SET hive.auto.convert.join.noconditionaltask = true;

SET hive.auto.convert.join.noconditionaltask.size = 10000000;

**--Skew Join**

SET hive.optimize.skewjoin=true;

SET hive.skewjoin.key=100000;

**--Use Cases 1**

**Landing and Exit Page**

SELECT edate, landPage, exitPage, COUNT(DISTINCT sessionid) as total\_sessions

FROM ( SELECT sessionid, edate,

first\_value(page) over (partition by sessionid) as landPage,

last\_value(page) over (partition by sessionid) as exitPage

FROM ( SELECT page, edate, sessionid, visitedtime,

count(\*) over(PARTITION BY sessionid order by visitedtime asc) as c,

rank() over(PARTITION BY sessionid order by visitedtime asc) as r

FROM customer\_action ) a

WHERE r = 1 or r = c ) b

GROUP BY edate, landPage, exitPage;