**Design document**

In order to solve the give assignment, I followed below design approach.

1. framework used : **HIVE (Hadoop framework)**
2. create two tables in hive
   1. calendar: created table by using below command
      1. CREATE TABLE calendar (datefield DATE);
      2. Inserted data by finding MIN & MAX date from given iput.
         1. Query to find MIN Date: select min(DATE(da)) from challenge
         2. Query to find MAX Date: select max(DATE(da)) from challenge
         3. Sample Data which got inserted into calendar table.

INSERT INTO calendar VALUES(‘2016-07-01’);

INSERT INTO calendar VALUES(‘2016-07-02’);

INSERT INTO calendar VALUES(‘2016-07-03’);

INSERT INTO calendar VALUES(‘2016-07-04’);

INSERT INTO calendar VALUES(‘2016-07-05’);

* 1. Cha: crated table to store the given input data.
     1. Da column stores date, type stores type and value stores value.
     2. CREATE TABLE cha (da STRING, type STRING, value int)

row format delimited

fields terminated by '\;'

LINES TERMINATED BY '\n';

* + 1. To load data: cha.csv file contains given input data.
       1. LOAD DATA INPATH '/user/cloudera/cha.csv'

INTO TABLE cha;

1. Design of query to produce results:
   1. I used three inner/sub queries and custom UDF (written in java) to produce the results.
   2. First inner query: Query calculates sum of all values with group by date and type

select da,type,cast(sum(value) as string) as ValuesSum from cha

group by da,type

order by da,type desc

**output:**

2016-07-01 impressions 1100

2016-07-01 conversions 1

2016-07-01 clicks 20

2016-07-02 impressions 20

2016-07-02 clicks 2

2016-07-04 impressions 10

* 1. Second inner query: this query takes output data of First inner query and performs RIGHT OUTER JOIN on **calender** table so that we get if there are any missing dates in given data.

SELECT calendar.datefield AS da, x.type, x.ValuesSum

FROM

(select da,type,cast(sum(value) as string) as ValuesSum from cha

group by da,type

order by da,type desc) x RIGHT JOIN calendar ON (DATE(x.da) = calendar.datefield)

**OUTPUT:**

2016-07-01 impressions 1100

2016-07-01 conversions 1

2016-07-01 clicks 20

2016-07-02 impressions 20

2016-07-02 clicks 2

2016-07-03 NULL NULL

2016-07-04 impressions 10

2016-07-05 NULL NULL

* 1. Outer query: It produces the actual results by using custom UDF which is written in java.

select result(concat(da,"#",concat\_ws("\;",collect\_list(type)),"#",concat\_ws("\;",collect\_list(ValuesSum))))

from

(SELECT calendar.datefield AS da, x.type, x.ValuesSum

FROM

(select da,type,cast(sum(value) as string) as ValuesSum from cha

group by da,type

order by da,type desc) x RIGHT JOIN calendar ON (DATE(x.da) = calendar.datefield))y

group by da;

**OUTPUT:**

2016-07-01;1100;20;1

2016-07-02;20;2;null

2016-07-03;null;null;null

2016-07-04;10;null;null

2016-07-05;null;null;null

**CUSTOM UDF:**

I have written custom UDF in java to give the proper results.

Input to custom UDF is:

2016-07-01#impressions;conversions;clicks#1100;1;20

2016-07-02#impressions;clicks#20;2

2016-07-03##

2016-07-04#impressions#10

2016-07-05##

2016-07-06##

For UDF, each row will be send as input at a time and this row will be processed and shown as result.

**Design of UDF:**

As soon as the one row as input to UDF function,

1. Converts row to string.
2. Split the string with delimiter “#”
3. First element in string array is date
4. Second element contains type
5. Third element contains value.
6. If Row length is 1 then will simply return “<Date>;null;null;null”
7. If second and third elements length is 3 the will return “<Date>;<impression value>;<Click value>;< conversions value>”
8. If row contains
   1. Only impressions
      1. Return “<Date>;<impression value>;null;null”
   2. Only clicks
      1. Return “<Date>;null;<Click value>;null”
   3. Only conversions
      1. Return “<Date>;null null;< conversions value>”
   4. Both impressions & clicks
      1. Return “<Date>;<impression value>;<clicks>;null”
   5. Both impressions & conversions
      1. Return “<Date>;<impression value>;null;< conversions >”
   6. Both clicks & conversions
      1. Return “<Date>;null;< clicks value>;< conversions value>”

**Implementation:**

package com.ca.testUDF;

import org.apache.hadoop.hive.ql.exec.UDF;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

public class Assignment extends UDF{

String result = "";

public Text evaluate(final Text s) {

if (s == null) { return null; }

String row = s.toString();

String[] rowValue = row.split("#");

//Date

String rowDate = rowValue[0];

if(row.split("#").length == 1) {

return new Text(rowDate + ";null;null;null");

}

//Table heading impressions;conversions;clicks

String[] type = rowValue[1].split(";");

if((rowValue[1].split(";").length == 3) && (rowValue[2].split(";").length == 3)) {

String[] valuesOfRow = rowValue[2].split(";");

return new Text(rowDate + ";" + valuesOfRow[0] + ";" + valuesOfRow[2] + ";" + valuesOfRow[1] );

}

String values = rowValue[2];

String[] rowValues = rowValue[2].split(";");

if(rowValue[1].contains("impressions") && !rowValue[1].contains("conversions") && !rowValue[1].contains("clicks")) {

values = values + ";null" + ";null";

}

if(rowValue[1].contains("conversions") && !rowValue[1].contains("impressions") && !rowValue[1].contains("clicks")) {

values = "null;" + ";null" + rowValues[0];

}

if(rowValue[1].contains("clicks") && !rowValue[1].contains("impressions") && !rowValue[1].contains("conversions")) {

values = "null;" + rowValues[0] + "null;" ;

}

if(rowValue[1].contains("impressions") && rowValue[1].contains("conversions")) {

String[] valuesOfRow = rowValue[2].split(";");

values = valuesOfRow[0] + ";null;" + valuesOfRow[1] ;

}

if(rowValue[1].contains("impressions") && rowValue[1].contains("clicks")) {

values = rowValues[0]+ ";" + rowValues[1] + ";null";

}

if(rowValue[1].contains("conversions") && rowValue[1].contains("clicks")) {

String[] valuesOfRow = rowValue[2].split(";");

values = "null;" + valuesOfRow[1] + ";" +valuesOfRow[0];

}

result = rowDate + ";" + values;

return new Text(result);

}

}

**TEST CASES:**

1. Check the date in input file once it is uploaded in HDFS.
2. Check tables (both calender & cha) in hive using HUE framework after creation of table.
3. Query content of calender table.
4. Query content of cha table.
5. Run complete query to check whether results are correct or not.
6. Check results of inner queries.
7. Check the logs for any issues.
8. Check output correctness for different fields.
9. Check output for missing dates.
10. Check output for delimiter between fields.