```
# s3939713
# Wing Hang Chan
# COSC 2637/2633 Big Data Processing
# COSC 2637/2633 Big Data Processing Assignment 2 ,Äì Handling Big Data
with Apache Pig

## Initialization
1.) Assume the 2 csv files are in HDFS and placed into /input
2.) Assume there are no /output/taskX folders

unzip s3939713_BDP_A2.zip
cd ./s3939713_BDP_A2/

Can run below commands to reset. Assume csv files placed in parent
directory.
hadoop fs -rm -f -r /input
hadoop fs -mkdir /input
hadoop fs -put ../cust_order.csv /input/cust_order.csv
hadoop fs -put ../order_line.csv /input/order_line.csv
hadoop fs -rm -f -r /output/task1
hadoop fs -rm -f -r /output/task2

## Run .pig script in hadoop master node
### Task 1
pig -x mapreduce task1.pig

### Task 2
pig -x mapreduce task2.pig
/**
    Load cust_order.csv
    i.e. select * FROM cust_order co;
*/
cust_order_header = LOAD 'hdfs:///input/cust_order.csv'
--cust_order_header = LOAD 'cust_order.csv'
    using PigStorage(',') as (
        order_id:int,
        order_datetime:chararray,
        customer_id:int,
        shipping_method_id:int,
        dest_address_id:int
        );

/**
    remove cust_order.csv header line
*/
cust_order_all = FILTER cust_order_header BY order_datetime !=
'order_date';
--cust_order_all_limit = LIMIT cust_order_all 10;
--dump cust_order_all_limit;

/**
    change datatype of order_datetime from chararray to date with format
    "yyyy-MM-dd HH:mm:ss"
*/
cust_order = FOREACH cust_order_all GENERATE
    order_id,
    ToDate(order_datetime, '"yyyy-MM-dd HH:mm:ss"') AS order_date;
--cust_order_limit = LIMIT cust_order 10;
```

```
--dump cust_order_limit;

/**
    Load order_line.csv
    i.e. select * FROM order_line ol;
*/
order_line_header = LOAD 'hdfs:///input/order_line.csv'
--order_line_header = LOAD 'order_line.csv'
       USING PigStorage(',') as (
       line_id:int,
       order_id:int,
       book_id:int,
       price:float
);

/**
    remove order_line.csv header line
*/
order_line = FILTER order_line_header BY line_id is not null;
--order_line_limit = LIMIT order_line 10;
--dump order_line_limit;

/**
    join cust_order and order_line with order_id
    i.e. FROM cust_order co
            INNER JOIN order_line ol ON co.order_id = ol.order_id
*/
cust_join_order = JOIN cust_order BY order_id, order_line BY order_id;
--cust_join_order_limit = LIMIT cust_join_order 10;
--dump cust_join_order_limit;

/**
    group by order_date (yyyy,M,dd) / (%Y %c %e) e.g. (2021,3,28)
    ** (%Y %c % e) is from discussion forum
    i.e. GROUP BY DATE_FORMAT(co.order_date, '%Y %c %e')
*/
group_order = GROUP cust_join_order BY ToString(order_date,
'(yyyy,M,d)');
--group_order_limit = LIMIT group_order 10;
--dump group_order_limit;

/**
    Construct bags as below:
    order date in format "(yyyy,M,d)"
    counting the number of book_id
    counting the unique number of order_id
    sum of the price
    i.e. DATE_FORMAT(co.order_date, '%Y-%m-%d') AS order_day,
        COUNT(DISTINCT co.order_id) AS num_orders,
        COUNT(ol.book_id) AS num_books,
        SUM(ol.price) AS total_price
*/
sum_data = FOREACH group_order {
    distinct_orders = DISTINCT cust_join_order.cust_order::order_id;
    GENERATE group as date,
    COUNT(cust_join_order.book_id) as num_books,
    COUNT(distinct_orders) as num_orders,
    SUM(cust_join_order.price) as total_price;
```

```
};
--sum_data_limit = LIMIT sum_data 10;
--dump sum_data_limit;

/**
    Order sum_data by price in desc order
    i.e. ORDER BY total_price DESC;
*/
order_sum_data = ORDER sum_data BY total_price DESC;
--sum_order_sum_data = LIMIT order_sum_data 10;
--dump order_sum_data_limit;

/**
    Store the result in HDFS
*/
STORE order_sum_data INTO 'hdfs:///output/task1';
--STORE order_sum_data INTO 'task1-results';
/**
    Load cust_order.csv
    i.e. select * FROM cust_order co;
*/
cust_order_header = LOAD 'hdfs:///input/cust_order.csv'
--cust_order_header = LOAD 'cust_order.csv'
    using PigStorage(',') as (
        order_id:int,
        order_datetime:chararray,
        customer_id:int,
        shipping_method_id:int,
        dest_address_id:int
        );

/**
    remove cust_order.csv header line
*/
cust_order_all = FILTER cust_order_header BY order_datetime !=
'order_date';
--cust_order_all_limit = LIMIT cust_order_all 10;
--dump cust_order_all_limit;

/**
    change datatype of order_datetime from chararray to date with format
    "yyyy-MM-dd HH:mm:ss"
*/
cust_order = FOREACH cust_order_all GENERATE
    order_id,
    ToDate(order_datetime, '"yyyy-MM-dd HH:mm:ss"') AS order_date;
--cust_order_limit = LIMIT cust_order 10;
--dump cust_order_limit;

/**
    Load order_line.csv
    i.e. select * FROM order_line ol;
*/
order_line_header = LOAD 'hdfs:///input/order_line.csv'
--order_line_header = LOAD 'order_line.csv'
       USING PigStorage(',') as (
       line_id:int,
       order_id:int,
       book_id:int,
       price:float
);

/**
    remove order_line.csv header line
*/
order_line = FILTER order_line_header BY line_id is not null;
--order_line_limit = LIMIT order_line 10;
--dump order_line_limit;

/**
    join cust_order and order_line with order_id
    i.e. FROM cust_order co
           INNER JOIN order_line ol ON co.order_id = ol.order_id
*/
cust_join_order = JOIN cust_order BY order_id, order_line BY order_id;
--cust_join_order_limit = LIMIT cust_join_order 10;
--dump cust_join_order_limit;

/**
    group by order_date (yyyy,M,dd) / (%Y %c %e) e.g. (2021,3,28)
    ** (%Y %c % e) is from discussion forum
    i.e. GROUP BY DATE_FORMAT(co.order_date, '%Y %c %e')
*/
group_order = GROUP cust_join_order BY ToString(order_date,
'(yyyy,M,d)');
--group_order_limit = LIMIT group_order 10;
--dump group_order_limit;

/**
    Construct bags as below:
    order date in format "(yyyy,M,d)"
    counting the number of book_id
    counting the unique number of order_id
    sum of the price
    i.e. DATE_FORMAT(co.order_date, '%Y-%m-%d') AS order_day,
        COUNT(DISTINCT co.order_id) AS num_orders,
        COUNT(ol.book_id) AS num_books,
        SUM(ol.price) AS total_price
*/
sum_data = FOREACH group_order {
    distinct_orders = DISTINCT cust_join_order.cust_order::order_id;
    GENERATE group as date,
    COUNT(cust_join_order.book_id) as num_books,
    COUNT(distinct_orders) as num_orders,
    SUM(cust_join_order.price) as total_price;
};
--sum_data_limit = LIMIT sum_data 10;
--dump sum_data_limit;

/**
    Define task2_udf.py as myfuncs in PIG
    Pass total_price to myfuncs.price_value and
    having a string "high value", "medium" or "low value" in return as
value
*/
Register 'task2_udf.py' using streaming_python as myfuncs;
```

```
sum_data_value = FOREACH sum_data
    GENERATE
    date,
    num_books,
    num_orders,
    total_price,
    myfuncs.price_value(total_price) as value;
--sum_data_value_limit = LIMIT sum_data_value 10;
--dump sum_data_value_limit;

/**
    Order sum_data by price in desc order
    i.e. ORDER BY total_price DESC;
*/
order_sum_data_value = ORDER sum_data_value BY total_price DESC;
--order_sum_data_value_limit = LIMIT order_sum_data_value 10;
--dump order_sum_data_value_limit;

/**
    Store the result in HDFS
*/
STORE order_sum_data_value INTO 'hdfs:///output/task2';
--STORE order_sum_data_value INTO 'task2-results';
from pig_util import outputSchema

# return a string called price_value to PIG scirpt
@outputSchema("price_value:chararray")
def price_value(num):
    if num < 100:
        # price lower than 100 is low value
        return "low value"
    if num < 300:
        # price higher or equal to 100 and lower than 300 is medium
        return "medium"
    # price higher or equal to 300 is high value
    return "high value"
```