1. Load the following datasets

```
divs = load '/pigData/NYSE_dividends' as (exchange:chararray,
symbol:chararray,date:chararray, dividends:float);
daily = load '/pigData/NYSE_daily' as (exchange:chararray,
symbol:chararray, date:chararray, open:float, high:float, low:float,
close:float, volume:int, adj_close:float);
-- The following also works, untyped
--divs = load '/pigData/NYSE_dividends' as (exchange, symbol, date,
dividends);
--daily = load '/pigData/NYSE_daily' as (exchange, symbol, date, open,
high, low, close, volume, adj_close);
players = load '/pigData/baseball' as (name:chararray, team:chararray,
position:bag{t:(p:chararray)}, bat:map[]);
crawl = load '/pigData/webcrawl' as (url, pageid, outpages:bag{t:
(p:chararray)} );
```

Note: These files can also be downloaded from http://www.utdallas.edu/~axn112530/cs6350/pigData/

2. From the NYSE_daily file, find the difference between close and open price for each stock for each day

```
   divs = load '/pigData/NYSE_dividends' as (exchange:chararray,
symbol:chararray,date:chararray, dividends:float);
   divs = load '/pigData/NYSE_dividends' as (exchange:chararray,
symbol:chararray,date:chararray, dividends:float);
   DESCRIBE divs;
   daily = load '/pigData/NYSE_daily' as (exchange:chararray,
symbol:chararray, date:chararray, open:float, high:float, low:float,
close:float, volume:int, adj_close:float);
   DESCRIBE daily;

   diff = foreach daily GENERATE symbol, data, (close-open);  (*:
Sourcecode)
   describe diff;
   dump;

   diff = foreach daily GENERATE symbol, date, (close-open) AS
diffPrice;
   describe diff;
   dump;
```

3. From the NYSE_dividends file, find the total dividends paid by each company so far

```
   o1 = foreach divs generate symbol,dividends;
   o2 = group o1 by symbol;
   describe o2;

   o3 = foreach o2 generate group, SUM(o1.dividends);
```

```
   describe o3;
   dump;

   o3 = foreach o2 generate group, SUM(o1.dividends) as sumOFDividens;
   describe o3;
   dump;

   o4 = order o3 by sumOFDividens desc;
   describe o3;

   dump;
```

4. From the NYSE_dividends file, find the total average dividend paid by those companies whose name starts with CM

```
   startswithcm = filter divs by symbol matches 'CM.*';
   describe startswithcm;
   o2 = group startswithcm by symbol;
   describe o2;
   o3 = foreach o2 generate group, AVG(startswithcm.dividends) as
avgDiv;
   dump o3;
   store o3 into 'pigOut';
   ls pigOut;
```

5. From the NYSE_daily, generate a listing of all the unique stocks

```
   o1 = foreach daily generate symbol;;
   describe o1;
   o2 = distinct o1;
   describe o2;
   o3 = order o2 by symbol;
   describe o3;
   dump o3;
```

6. Join the two tables on the symbol key and then filter the joined relation by those stocks whose closing price is > 10. Then create a random 10% sample of the output and display on screen.

```
   jnd = join divs by symbol, daily by symbol;
   describe jnd;

   filtrd = filter jnd by daily::close > 10;
   describe filtrd;

   smpled = sample filtrd 0.1;
   describe smpled;

   dump;
   store smpled into 'fltrdDivs';
```

7. For the NYSE_daily dataset group by the symbol and find the average daily close for each group. Then order the data using average closing price in a descending way.
You have to use a 10 reducers for the average calculation and 2 reducers for sorting.

```
   grpd = group daily by symbol;
   describe grpd;

   avgClose = foreach grpd generate group, AVG(daily.close) parallel
10;
   dump;

   avgClose = foreach grpd generate group, AVG(daily.close) as
closePrice parallel 10;
   describe avgClose;

   avgClose1 = order avgClose by closePrice desc parallel 2;
   dump avgClose1;
```

8. Register the piggybank.jar file from '/usr/local/pig-0.13.0/lib/piggybank.jar' and then generate the symbol and reverse of every symbol

```
   register '/usr/local/pig-0.13.0/lib/piggybank.jar'
   backwards = foreach divs generate symbol,
org.apache.pig.piggybank.evaluation.string.Reverse(symbol);
   dump backwards;
```

9. Look at the baseball dataset. Use describe command. What do you think is the datatype of the position field. Flatten this dataset and output the name of player and number of positions in which he has played. The output should be ordered in descending order of number of positions.

```
   players = load '/pigData/baseball' as (name:chararray,
team:chararray, position:bag{t:(p:chararray)}, bat:map[]);
   dump players;

   (bag datatype -> flatten bag datatype)
   fltnd = foreach players generate name, flatten(position) as p;
   describe fltnd;
   dump fltnd;

   grpd = group fltnd by name;
   cntd = foreach grpd generate group, COUNT(fltnd.p);
   dump cntd;
```

10. Using the baseball dataset, first flatten the positions column and then generate a listing of the count of players for each position in a

```
descending order.
    (fltnd = foreach players generate name, flatten(position) as p;
    grp1 = group fltnd by p;
    grp2 = foreach grp1 generate group, COUNT(fltnd.name);)

    fltnd = foreach players generate name, flatten(position) as pos;
    grp1 = group fltnd by pos;
    grp2 = foreach grp1 generate group, COUNT(fltnd.name);
    dump grp2;

11. Group the NYSE_daily data by exchange and for each exchange find
the count of symbols.
    grpd = group daily by exchange;
    uniqcnt = foreach grpd {sym = daily.symbol; uniq_sym = distinct
sym; generate group, COUNT(uniq_sym);};
    dump uniqcnt;


12. The number of entries in the divs table is less than daily table.
Join them using a fragment replicated join. Check to see if a reduce
phase is needed.
    jnd = join daily by (exchange, symbol), divs by (exchange, symbol)
using 'replicated';
    dump jnd;

13. Create a co-group of daily and divs relations and find out those
symbols that have never paid a dividend.
    grpd = cogroup daily by(exchange, symbol), divs by
(exchange,symbol);
    sjnd = filter grpd by IsEmpty(divs);
    final = foreach sjnd generate flatten(daily);
    describe final;
    final1 = foreach final generate final.daily::symbol as sym;
    final2 = distinct final1;
    dump final2;

14. Use the crawl data and find for each page the number of outlinks
and number of inlinks.
    crawl = load '/pigData/webcrawl' as (url, pageid, outpages:bag{t:
(p:chararray)} );
    fltnd = foreach crawl generate url, flatten(outpages.t) as
outlinks;
    describe fltnd;
    grpd = group fltnd by url;
    cntOut = foreach grpd generate group, COUNT(fltnd.outlinks) as
numOut;
```