COMP421 Assignment 3
Name: Edgar Chang
Student Id: 260729484

Q1.

a. For each friend F inside the input person H's friend list, output the key value pair (F, 1), where F is the name of the friend.
To compute the common friends of H and her friend F1, go to F1's friend list and output the key value pair (S,1) for each friend S of F1.
Sort all outputs with the same key into a key/value-list pair. For each key X with value count greater than 1, output (H, F1), (X). Finally group all outputs together to produce the common friend list as key/value-list pair (H, F1), (X,Y,Z,K...). We can follow the same steps for all friends of H.

b.
For each friend F inside the input person X's friend list, output the key value pair (F, 1), where F is the name of the friend.
To compute the common friends of X and Y, go to Y's friend list and output the key value pair (S,1) for each friend S of Y.
Sort all outputs with the same key into a key/value-list pair. For each key A with value count greater than 1, output (X, Y), (A). Finally group all outputs together to produce the common friend list as key/value-list pair (X, Y), (A, B, C...).
Since the output of the workflow is a key/value-list pair, we can simply lookup the value-list with key (X,Y).

c. Yes this workflow allows practically duplicate key (X,Y) and (Y, X) to be stored at the same time. We can avoid this by looking up if there is already a list associated with the twin key. For example, when trying to generate a mutual friends list for (X, Y), look up to see if (Y, X) is already present with a value-list. If so then skip (X, Y). We need to modify the lookup strategy by looking up both the (X, Y) and (Y, X) keys.

d. If we input Joe, (Abe, Jane, Ali) to the first mapper, it should output (Abe, 1), (Jane, 1), (Ali, 1). Then we should input another person, say Jane, (Abe, Ana, Jude) to the mapper, which should produce (Abe, 1), (Ana, 1), (Jude, 1). Then we send all the outputs to the reducer, which will group outputs with the same key together and sum the value. We will get (Abe, 2), (Jane, 1), (Ali, 1), (Ana, 1), (Jude, 1). Then for each key with value greater than 1, we output (Joe, Jane), (Abe).

Q2.

For each record in Patient that has age above 60, a mapper outputs (Hname, 1).

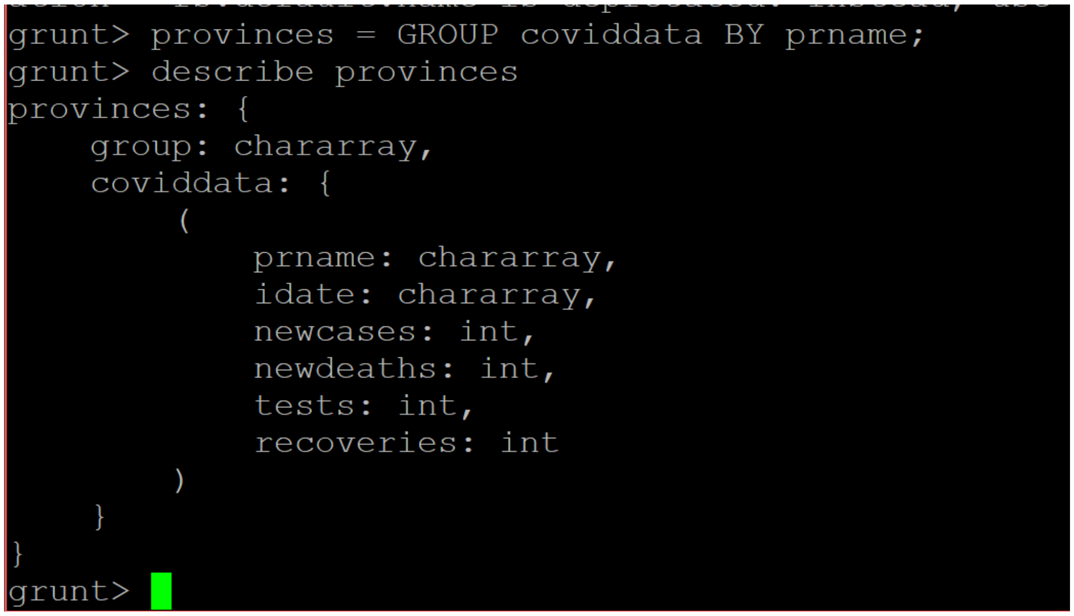
```
For each record (Hname, Province) in Hospital:
    For each output from previous mapper with Hname = Hname:
        Output (Province, 1)
```

Shuffle and group all output with same key into a kay/value-list pair (Province, (1,1,1,1...)).
Reduce and sum all the number in the value-list and output key-value pair (Province, number). Output the pair with number greater than 100.

For example, if we input Patient(123, 62, McGill General) into the first mapper, it should out (McGill General, 1). If the age is below 60, nothing should be outputted. Then for the hospital (McGill General, Quebec), we look at the previous outputs where the key is McGill General. For each of these outputs, it should output (Quebec, 1). Then we group all outputs with key Quebec into a key/value-list tuple like (Quebec, (1,1,1,1,1,....)). Then we sum up the value list and output the tuple with sum greater than 100, like (Quebec, 302).

Q5.

C.



```
grunt> provinces = GROUP coviddata BY prname;
grunt> describe provinces
provinces: {
    group: chararray,
    coviddata: {
        (
            prname: chararray,
            idate: chararray,
            newcases: int,
            newdeaths: int,
            tests: int,
            recoveries: int
        )
    }
}
grunt> 
```

Screenshot of schema after grouping

Q6.

C.

I did not use a join operation in this question, which the professor said was allowed as long as the final output is correct.