1. Design a program in your favorite programming language that can generate tuples where field values can be unique, have a fixed set of distinct values that are allocated to tuples according to a uniform probability distribution, and a fixed set allocated to the tuples according to a fractal probability distribution (70-30 rule). An example function is pseudocode to do this is as follows:

   gen(frac, N)
   begin
     p:= random permutation of numbers from 1 to N
     outvec:= p // so outvec is of length N
     while(|p| > 1
       p:= first frac*|p| elements of p
       concatenate p to outvec
     end while
     return random permutation of outvec
   
   Execute this program with gen(0.3, x) where x is 70,000 or so (might be more). The idea is to generate an array of 100,000 fractally distributed numbers.

   Populate table sales(id, itemid, quantity) such that there are 100,000 ids, 20,000 itemids distributed with uniform probability among the sale ids. Then quantities are to be distributed in a fractal manner by using the above-generated fractal array.

   Next write one or more queries that find the ten items having the greatest total quantities (sum of the quantities grouped by item) and the ten items having the least total quantities.

   Use any relational database on any operating system you like. Show us the queries and then a terminal history session showing execution and the results.

2. Choose two rules of thumb from the book (e.g., rules having to do with chopping, indexes, commits, checkpoints, etc.). For each one, find (and demonstrate using mysql or some other database system) data distributions/access patterns where it is satisfied and data distributions/access patterns where it is not. A data distribution refers to the frequency of occurrences of different values of a given variable across the rows - e.g. uniform and fractal are two different (and convenient since you already generated data with them) distributions. Thus, you need to show four different performance numbers for each rule of thumb derived by executing real queries in a real database: results with and without the rule of thumb applied to columns.
with two different data distributions. Comment on whether the case (i.e. the data distribution) where the rule of thumb is satisfied is more likely than when it isn’t. Also explore how you would modify the statement of the rule of thumb to make it more precise given the performance results.

3. ReserveWithUs is an electronic portal for hotel rooms. It buys rooms at discounted rates and sells them for a bit more (see hotels.com for a real-life model of such a company). The company is composed of a purchasing department that negotiates deals with hotels and a sales department that monitors customer activities. Each hotel has a few types of room (e.g., superior, king, queen). For each type of room in a given hotel, the purchasing department negotiates a block of rooms of a given type at a given price on a given day. Potential customers browse the web site to find attractive hotels and room deals. Possibly, they log in and add a (number of) rooms to their shopping cart. Customers can then buy all the rooms in their shopping cart, a so-called shopping cart checkout, where the information about the booked rooms are committed to the database and archived. You are the IT person at ReserveWithUs and you have been given a simple application server. The application server is written in java (tested on java 1.6). The application server together with the database schema and scripts to generate data is available at https://databasetuning-cases.googlecode.com/files/ReserveWithUs-2.3.zip The design of the app server is described on that web site. The source code is available and you should take some time to familiarize yourself with it.

Your job is to tune the database portion of the application in your environment and to redesign some portions of the application server that are poorly designed. Specifically, your project is to document the following tasks (back up your claims with arguments where possible with quantitative arguments based on experiments):

a) No index is defined in the database. You should pick the appropriate indexes assuming that the workload has the following characteristics (note that you should give arguments for your choice based on the nature of the workload and on the nature of the queries on the website).

b) Make the following queries fast:
   i) query on hotel and room based on country and city
   ii) queries on rooms contain a date interval
   iii) export information about all customers, delete all customers and import the information again.
   iv) checkout.

As you do all this work, please keep a log of the problems you have faced. Demonstrate the effect of all the changes you have made with performance tables and graphs.