DBMS HOMEWORK #1

Important Note:

- At the begining of your answer sheet, **note your Postgresql version** that you used for this homework.
- At the answers, **show your work**, SQL, comments on EXPLAIN output and simple calculations i.e.
- You are expected to work **in a group of at most 2.**

(1)

Define a table "numbers" with 3 attributes as below.
numbers (**id** *integer*, **A** *integer*, **B** *integer*)
"numbers" table are expected to store 2M (2 million) tuples. Data distribution of each attribute is as below:

- **id** values between 1-1M
- **A** values between 1-1M (high cardinality)
- **B** values between 1-1K (low cardinality)

Test 1:
First, Generate & load data for this table.
Second, define & build B+-tree indexes a_idx on **A**, b_idx and **B** columns.

Test 2:
First, define a B+-tree indexes "a_idx on **A**" and "b_idx on **B**" columns.
Second,Generate & load 2M (2 million) tuples 1-by-1 for this table.

For each test above, Evaluate and **compare** the following metrics. Convey your reasoning on the latency values.

(a) data+indexes loading time :

- You may as well use \timing service in psql.
- For Test1, add 3 distinct latencies, "dataload latency"+"a_idx latency"+ "b_idx latency".

	Loading Latency
Test 1	
Test 2	

(b) index sizes on disk of "numbers" table & a_idx and b_idx:

- both from each Test above and both a-idx and b_idx. Write your comments if they differ between tests or between 2 indexes.

	numbers heap file(w/o idx)	a_idx	b_idx
Test 1			
Test 2			

(c) index statistics. ("space utilization" and "tree height"?)

	a_idx	b_idx
Test 1		
Test 2		

(d) Based on the previous evaluations, which scenario is preferable?
First load data table and then define&load indexes? OR define table&indexes and then load data?

(2)

"**analyze**" is an admin command in databases to refresh the table stats.

(a) **When** and **why** do we use this command. **What** operations are done internally? Explain briefly. (at most 100 words)

(b) Load the previously defined "numbers" table with 1 M tuples again.

- Test 1: First load and then display the statistics for each attribute (i.e. number of distinct tuples, most_common_values&frequencies)
- Test 2: First load & **analyze** and then display the statistics for each attribute (i.e. number of distinct tuples, most_common_values&frequencies)

Write the meaning of values of the stat's output briefly.
(you may as well use pg_stats utility for the statistics.)

(3)

Load the previously defined "numbers" table with 1 M tuples again. Load a-idx & b_idx as well.
We want to display tuples (with all attributes) sorted by "val".

(a) Run the query & "explain" the explain output.

(b) Most probably you are seeing **external sorting** !. Now increase "required buffer area" until you see "quick-sorting in main memory".

If you are already seeing **quick-sorting**, decrease required buffer area until you see "external-sorting in main memory".

(c) Have you experinced better execution times when you increased the buffer size? Why or why not?

(4)

Load the previously defined "numbers" table with 1 M tuples again. Load a-idx & b_idx as well.

(a) **We want to count distinct "a" values.** Run the query & "explain" the explain output. Why does it use (or not use) idx ?

(b) **We want to list all tuples (with all attributes) having a B-value higher than 700.** Run the query & "explain" the explain output. Why does it use (or not use) idx ? What is the threshold value to see sequential file scan?

(c) **We want to count the total number of A-values that are equal to B-values in "any tuple in the table", including duplicates.** Run the query & "explain" the explain output. Why does it use (or not use) idx ?