EX 3

2.a) The ChatGpt query computes all of the Donors tuples with different names, so there exists at least one organization both donors donated to.

This happens because after using the "WHERE" keyword, there will remain only rows where there are Donors tuples with different names, and the same organization they donated to. Then, when we subgroup by name, the "HAVING" condition will certainly exist – because we indeed left the rows with the same organization for different donors. i.e., each donor will have the same organizations as the second donor, so the amount will be equal.

---

1. a) We will have to check the entire table. There are 1000 rows in the table, each row is 64 bytes sized. Therefore, the entire table byte size is 64000. Each block size is 1024 bytes, so the total amount of blocks in the table is ceil(64000/1024) = 63. Therefore the total cost of computing the table is 63.

b). Let us assume the maximum nesting level is d. each pointer takes 4 bytes, so all pointers take 4d bytes. "Birthyear" which is a key, takes 4 bytes, and there are (d-1) of those, so total of 4(d-1).

$4d + 4(d-1) <= 1024$, so $d = 128$.

c). Let us assume the nesting level is 128. The cost of traversing the tree, by the formula we learned would be 2.

Additionally, there is a cost of 1 more for the leaf, so in total 2 + 1 = 3.

2. Let us assume the nesting level is 128. We would want to traverse the tree and also reach all of the suitable leaves. The cost of traversing a tree is the same as the previous question, 2. Let us compute the cost of reaching the leaves. "birthyear" distributes uniformly on (1990, 2000]. So there are ((2000-1990)/100) * 1000 = 100 rows with suitable values in

them. Therefore the amount of relevant blocks is ceil(100/((128/2)-1)) = 2. So in total we'll get 2 + 2 = 4.

3. a) Let us assume that the maximum nesting level is d. a pointer size is 4 bytes, so there are 4d bytes reserved for pointers. "Uid" is the size of 4 bytes, there are (d-1) of those, so there are 4(d-1) bytes reserved for "Uid". So we'll get: 4d+ 4(d-1) <= 1024, therefore the maximum nesting level is 128.

b) Let us assume that the maximum nesting level is 128. The cost of traversing a tree, as we computed earlier is 2. Since "Uid" is a key, we need to visit exactly one leaf, so we'll add 1 to the cost. Additionally, we would have to access one row of the Table to extract the name, so we'll add 1 to the cost. So in total, 2 + 1 + 1 = 4.

4. a) Let us assume that the nesting level is d. a pointer is 4 bytes-sized, so there is 4d bytes reserved for pointers. "Language" is the size of 10 bytes, there are d-1 key values so there is 10(d-1) bytes reserved for that. So, well get that 4d+ 10(d-1) <= 1024, meaning that the maximum nesting level is 73.

b) let us assume that the nesting level is 73. According the formula we've learned, the cost of traversing this tree is 2. Now we'll compute the cost of visiting the leaves. From the data, the languages uniformly diverge to 5 so the number of relevant rows is 1000/5 = 2. So the number of relevant leaves is ceil(200/((ceil(73/2) – 1)) = 6. And that is the cost of visiting the leaves.

Let us compute the cost of accessing the table. The amount of relevant rows is 200, and they can be in different blocks, so well get 200 blocks. Reading all of the table costs 63 operations, so if we will read those 63 blocks, we will also read the entire 200rows. In total we will get: 2 + 6 + min(63, 100) = 71.

c) let us assume the maximum nesting level is d. a pointer size is 4 bytes, so we reserve 4d bytes for the pointers. The language size is 10 bytes, and "uid" size is 4. There are (d-1) key values, so there is 14(d-1) bytes reserved for them. In total we will get 4d + 14(d-1) <= 1024, so d = 57.

d) Let us assume that the maximum nesting level is 57. We will compute the cost of traversing the tree by the formula we learned(3). We will compute the cost of visiting the leaves. As before, there are 200 relevant rows. So the cost of visiting the leaves is ceil(200/(ceil(57/2)-1)) = 8.

In total we will get: 8 + 3 = 11.