Homework 2

Bingying Liang Due: Oct. (Late deadline: Dec 2nd 11:59pm.)

1. Consider the following query

```sql
SELECT *
FROM A, B
WHERE A.x = B.x
```

Calculate (or provide lower and/or upper bound, if you don't have enough information) the number of tuples returned from the query, under the following conditions (each part is separate from the other). Unless otherwise stated, table A has 10000 tuples, table B has 20000 tuples.

a. A.x is the primary key of x, and B.x is the primary key of B

b. A.x is the primary key of x, and B.x is the foreign key refer to A.x (but B.x is not unqiue)

c. A.x and B.x is evenly distributed between 1-100.

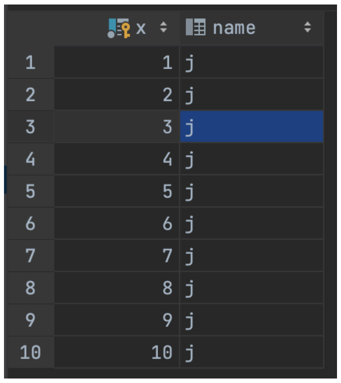
d. A.x is evenly distributed between 1-100, B.x is evenly distributed between 51-150.

Solution:

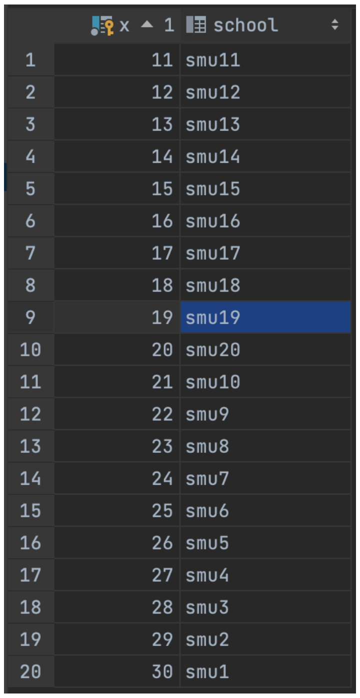
- **a.** Because the x is the primary key of both A and B, which means there is no duplicate tuples on the x.
  - **1 The lower bound:**

    Unlucky: None of 10000 tuples of A.x is equal to B.x

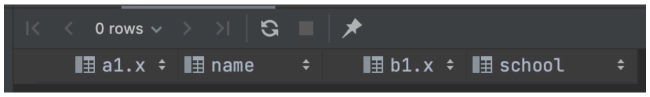
    Therefore return 0 tuples



	x	name
1	1	j
2	2	j
3	3	j
4	4	j
5	5	j
6	6	j
7	7	j
8	8	j
9	9	j
10	10	j



	x	school
1	11	smu11
2	12	smu12
3	13	smu13
4	14	smu14
5	15	smu15
6	16	smu16
7	17	smu17
8	18	smu18
9	19	smu19
10	20	smu20
11	21	smu10
12	22	smu9
13	23	smu8
14	24	smu7
15	25	smu6
16	26	smu5
17	27	smu4
18	28	smu3
19	29	smu2
20	30	smu1

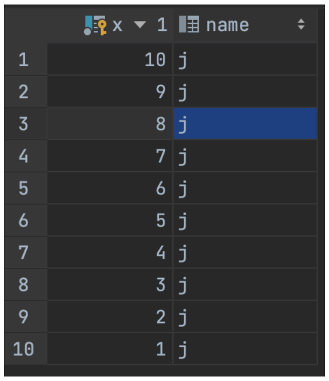


a1.x	name	b1.x	school

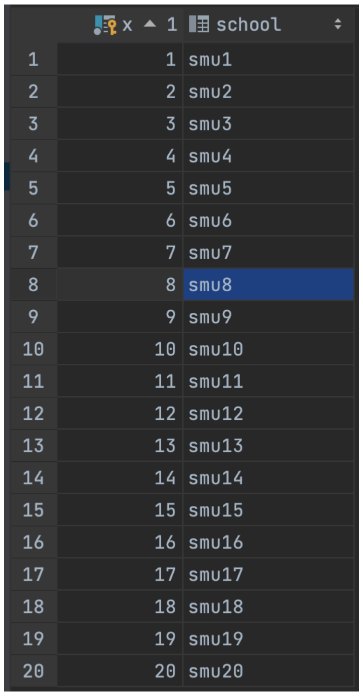
- **2 The upper bound:**

  Lucky: Every tuples of A.x is equal to B.x and none of them is duplicated.

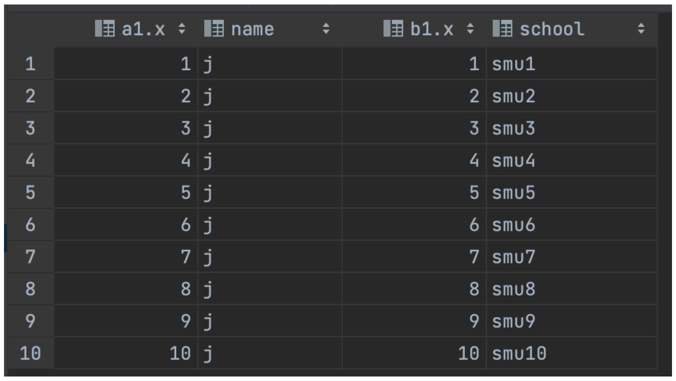
  Therefore return 10000 tuples.



	x	name
1	10	j
2	9	j
3	8	j
4	7	j
5	6	j
6	5	j
7	4	j
8	3	j
9	2	j
10	1	j



	x	school
1	1	smu1
2	2	smu2
3	3	smu3
4	4	smu4
5	5	smu5
6	6	smu6
7	7	smu7
8	8	smu8
9	9	smu9
10	10	smu10
11	11	smu11
12	12	smu12
13	13	smu13
14	14	smu14
15	15	smu15
16	16	smu16
17	17	smu17
18	18	smu18
19	19	smu19
20	20	smu20



	a1.x	name	b1.x	school
1	1	j	1	smu1
2	2	j	2	smu2
3	3	j	3	smu3
4	4	j	4	smu4
5	5	j	5	smu5
6	6	j	6	smu6
7	7	j	7	smu7
8	8	j	8	smu8
9	9	j	9	smu9
10	10	j	10	smu10

Therefore the lower bound is 0, the upperbound is 10000.

- **b.** Because the x is the primary key of A, therefore, there is no duplicate tuples on the x of table A.

  But B.x is the foreign key refer to A.x, therefore, B every tuples reference to A, which means every B.

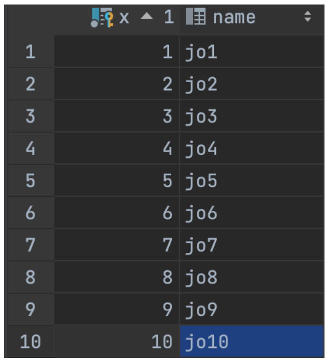
  tuple is matched with A.x .

  It will return the number B

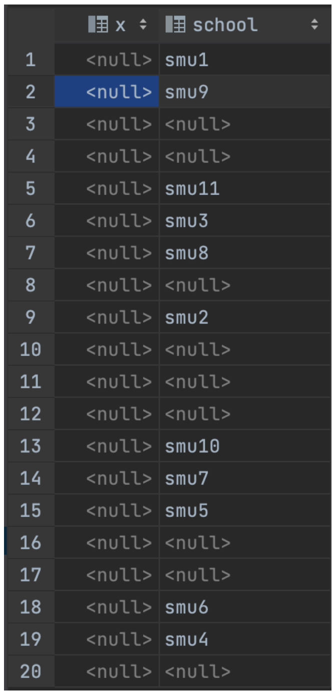
  - **1 The lower bound:**

    Unlock: Because B.x is not the primary key of the B. so it can be none. If all the B.x is none, therefore, 0 tuples will matches A.x = B.x.

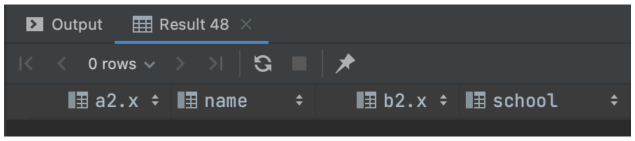
    Therefore, it return 0 tuples.



	x	name
1	1	jo1
2	2	jo2
3	3	jo3
4	4	jo4
5	5	jo5
6	6	jo6
7	7	jo7
8	8	jo8
9	9	jo9
10	10	jo10



	x	school
1	<null>	smu1
2	<null>	smu9
3	<null>	<null>
4	<null>	<null>
5	<null>	smu11
6	<null>	smu3
7	<null>	smu8
8	<null>	<null>
9	<null>	smu2
10	<null>	<null>
11	<null>	<null>
12	<null>	<null>
13	<null>	smu10
14	<null>	smu7
15	<null>	smu5
16	<null>	<null>
17	<null>	<null>
18	<null>	smu6
19	<null>	smu4
20	<null>	<null>

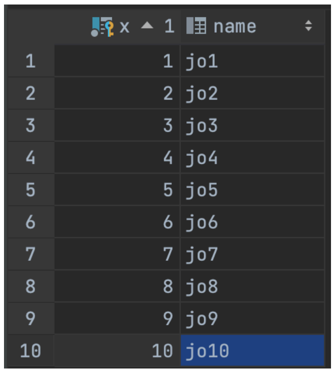


a2.x	name	b2.x	school

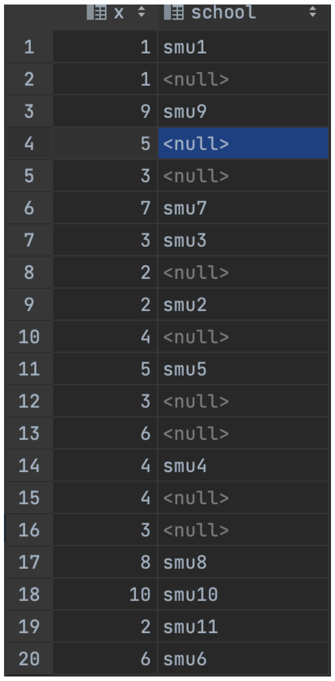
- **2 The upper bound:**

  Lucky: Every B.x is not none, so every tuples reference to the A.x, therefore, it can return the whole B tuples, which is 20000 tuples.

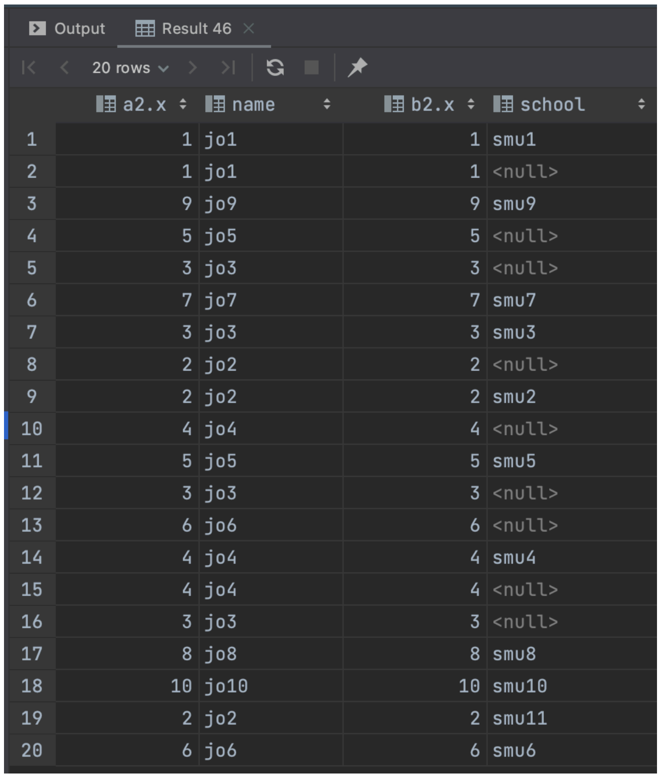
  Therefore, it return 20000 tuples.



	x	name
1	1	jo1
2	2	jo2
3	3	jo3
4	4	jo4
5	5	jo5
6	6	jo6
7	7	jo7
8	8	jo8
9	9	jo9
10	10	jo10



	x	school
1	1	smu1
2	1	<null>
3	9	smu9
4	5	<null>
5	3	<null>
6	7	smu7
7	3	smu3
8	2	<null>
9	2	smu2
10	4	<null>
11	5	smu5
12	3	<null>
13	6	<null>
14	4	smu4
15	4	<null>
16	3	<null>
17	8	smu8
18	10	smu10
19	2	smu11
20	6	smu6



	a2.x	name	b2.x	school
1	1	jo1	1	smu1
2	1	jo1	1	<null>
3	9	jo9	9	smu9
4	5	jo5	5	<null>
5	3	jo3	3	<null>
6	7	jo7	7	smu7
7	3	jo3	3	smu3
8	2	jo2	2	<null>
9	2	jo2	2	smu2
10	4	jo4	4	<null>
11	5	jo5	5	smu5
12	3	jo3	3	<null>
13	6	jo6	6	<null>
14	4	jo4	4	smu4
15	4	jo4	4	<null>
16	3	jo3	3	<null>
17	8	jo8	8	smu8
18	10	jo10	10	smu10
19	2	jo2	2	smu11
20	6	jo6	6	smu6

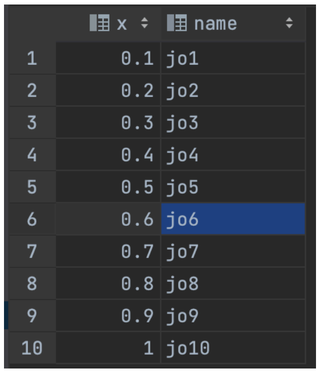
Therefore the lower bound is 0, the upperbound is 20000.

- **c.**
  - **1 The lower bound:**
    - A.x has 10000 tuples distribute evenly 1-100, therefore we can divided 1-100 as $\frac{100}{10000} = 0.01$, Suppose the first tuple of A.x that x = 0.01, and the second x = 0.02 ... the last one x = 100.00

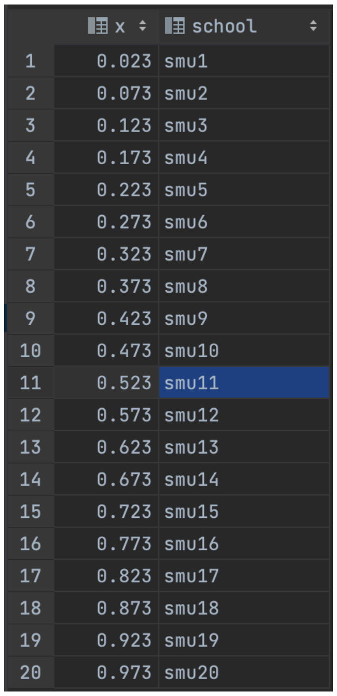
B do the same thing, divided 1-100 as 20000 parts, and each parts' range is $\frac{100}{20000} = 0.005$, supose the first tuple of x is 0.00023, the second x = 0.00523, ... the last one x = 19999.99523

Obvisouly , no tuple A.x = B.x can match.

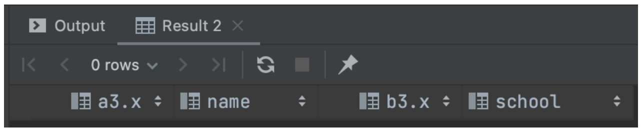
Therefore, return 0 tuples



	x	name
1	0.1	jo1
2	0.2	jo2
3	0.3	jo3
4	0.4	jo4
5	0.5	jo5
6	0.6	jo6
7	0.7	jo7
8	0.8	jo8
9	0.9	jo9
10	1	jo10



	x	school
1	0.023	smu1
2	0.073	smu2
3	0.123	smu3
4	0.173	smu4
5	0.223	smu5
6	0.273	smu6
7	0.323	smu7
8	0.373	smu8
9	0.423	smu9
10	0.473	smu10
11	0.523	smu11
12	0.573	smu12
13	0.623	smu13
14	0.673	smu14
15	0.723	smu15
16	0.773	smu16
17	0.823	smu17
18	0.873	smu18
19	0.923	smu19
20	0.973	smu20



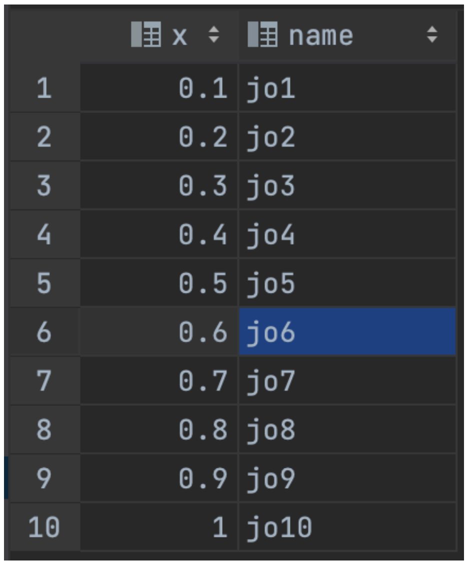
a3.x	name	b3.x	school

- **2 The upper bound:**
  - x is not the primay key of A and B, therefore, B.x or A.x can be dulicated. A.x has 10000 tuples, therefore we can divided 1-100 as $\frac{100}{10000} = 0.01$, suppose the first tuple of x is 0.01, the second tuple A.x = 0.02, ... the last one x =100.00

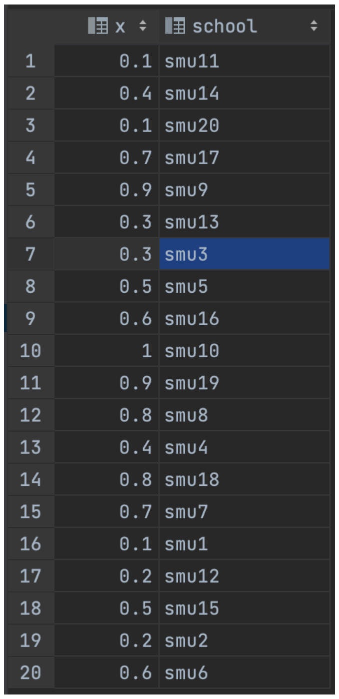
    Because B has 20000 tuples, which is 2 times of A.

    Now let B.x first tuple = 0.01 and second tuple of B.x =0.01, and the third and fouth tuple of B.x = 0.02 ... the 19999th and 20000 th tuple of B.x = 10000.00 .Therefore, every B table B.x mathes with the table A.x.

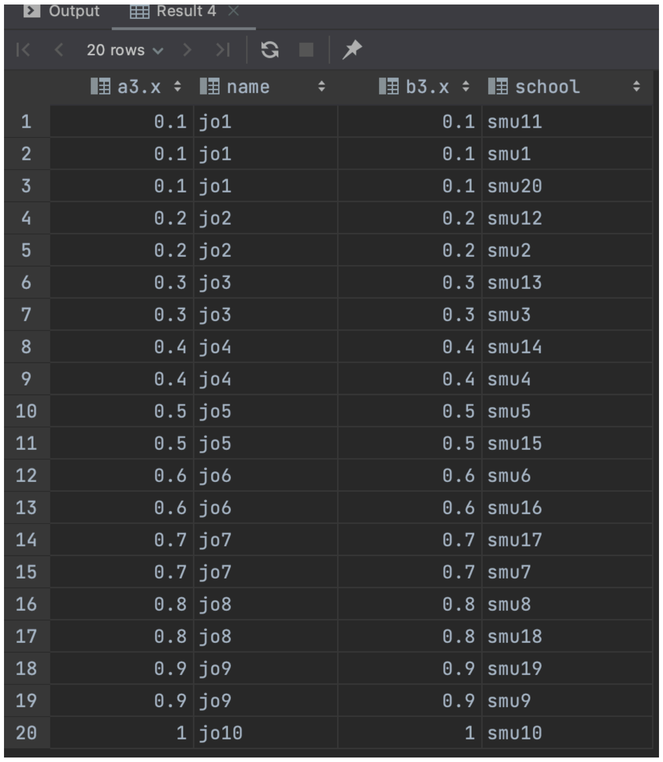
Therefore, return 20000 tuples.



	x	name
1	0.1	jo1
2	0.2	jo2
3	0.3	jo3
4	0.4	jo4
5	0.5	jo5
6	0.6	jo6
7	0.7	jo7
8	0.8	jo8
9	0.9	jo9
10	1	jo10



	x	school
1	0.1	smu11
2	0.4	smu14
3	0.1	smu20
4	0.7	smu17
5	0.9	smu9
6	0.3	smu13
7	0.3	smu3
8	0.5	smu5
9	0.6	smu16
10	1	smu10
11	0.9	smu19
12	0.8	smu8
13	0.4	smu4
14	0.8	smu18
15	0.7	smu7
16	0.1	smu1
17	0.2	smu12
18	0.5	smu15
19	0.2	smu2
20	0.6	smu6



	a3.x	name	b3.x	school
1	0.1	jo1	0.1	smu11
2	0.1	jo1	0.1	smu1
3	0.1	jo1	0.1	smu20
4	0.2	jo2	0.2	smu12
5	0.2	jo2	0.2	smu2
6	0.3	jo3	0.3	smu13
7	0.3	jo3	0.3	smu3
8	0.4	jo4	0.4	smu14
9	0.4	jo4	0.4	smu4
10	0.5	jo5	0.5	smu5
11	0.5	jo5	0.5	smu15
12	0.6	jo6	0.6	smu6
13	0.6	jo6	0.6	smu16
14	0.7	jo7	0.7	smu17
15	0.7	jo7	0.7	smu7
16	0.8	jo8	0.8	smu8
17	0.8	jo8	0.8	smu18
18	0.9	jo9	0.9	smu19
19	0.9	jo9	0.9	smu9
20	1	jo10	1	smu10

Therefore the lower bound is 0, the upperbound is 20000.

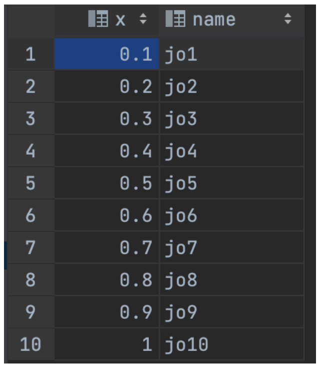
- **d.**
  - **1 The lower bound:**

    A.x has 10000 tuples, distribute evenly between 1-100, therefore we can divided range 100 into 10000 parts, each part range is $\frac{100}{10000} = 0.01$, let first tuple A.x = 0.01, and the second is x = 0.02... the last one x = 100.00

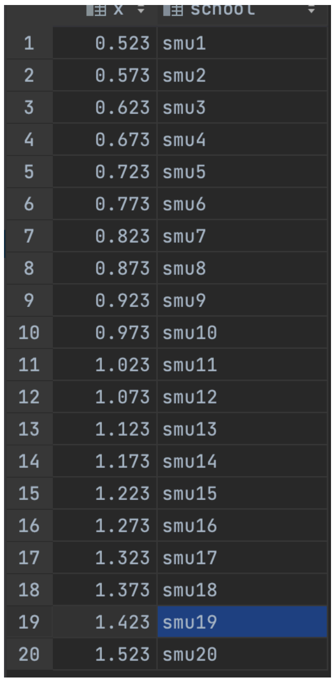
    B.x has 20000 tuples, distribute evenly between 51-150, therefore we can divided the range into 20000 parts, each part range is $\frac{100}{20000} = 0.005$, the first tuple of B, let's say B.x = 51.00023, the second is B.x = 51.00523... the last one B.x = 149.99523.

    In this situation, the there is no tuple satisfy that A.x = B.x, therefore return 0 tuples.

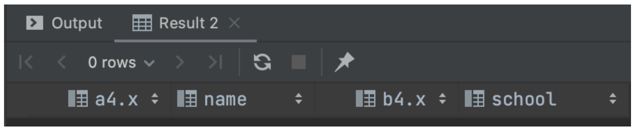
    Therefore, return 0 tuples



	x	name
1	0.1	jo1
2	0.2	jo2
3	0.3	jo3
4	0.4	jo4
5	0.5	jo5
6	0.6	jo6
7	0.7	jo7
8	0.8	jo8
9	0.9	jo9
10	1	jo10



	x	school
1	0.523	smu1
2	0.573	smu2
3	0.623	smu3
4	0.673	smu4
5	0.723	smu5
6	0.773	smu6
7	0.823	smu7
8	0.873	smu8
9	0.923	smu9
10	0.973	smu10
11	1.023	smu11
12	1.073	smu12
13	1.123	smu13
14	1.173	smu14
15	1.223	smu15
16	1.273	smu16
17	1.323	smu17
18	1.373	smu18
19	1.423	smu19
20	1.523	smu20



a4.x	name	b4.x	school

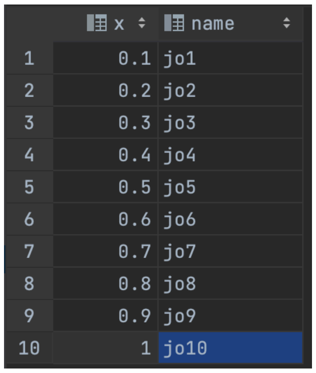
- **2 The upper bound:**

  Because A.x has 10000 tuples between 0-100 and distribute evenly, which means there are at most $\frac{10000}{2} = 5000$ tuples between 51-100.

  Because B.x has 20000 tuples between 51-150 and distribute evenly, which means there are at most $\frac{20000}{2} = 10000$ tuples between 51-100.

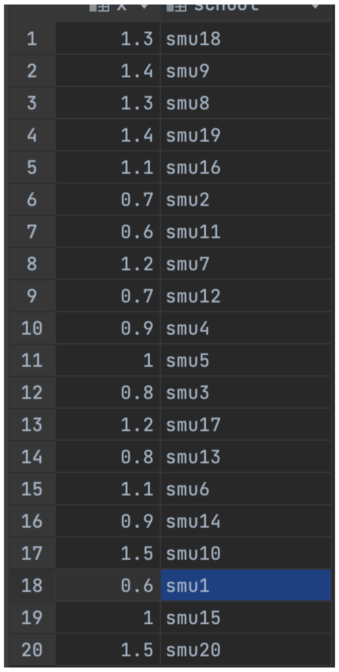
  Therefore, we can let 5000 tuples A.x = B.x, and the last 5000 tuples let duplicated to A.x. For example, divided 51-100 into 5000 parts , and each parts range is $\frac{50}{5000} = 0.01$ parts. If first A.x = 50.01, and the second A.x = 50.02 ... the 5000th A.x = 100. Now if 1st and 2nd B.x =50.01 and 3rd and 4th B.x = 50.02... the 9999th and 10000th B.x = 100. Therefore, there are 10000 tuples A.x = B.x

  Therefore return 10000 tuples.

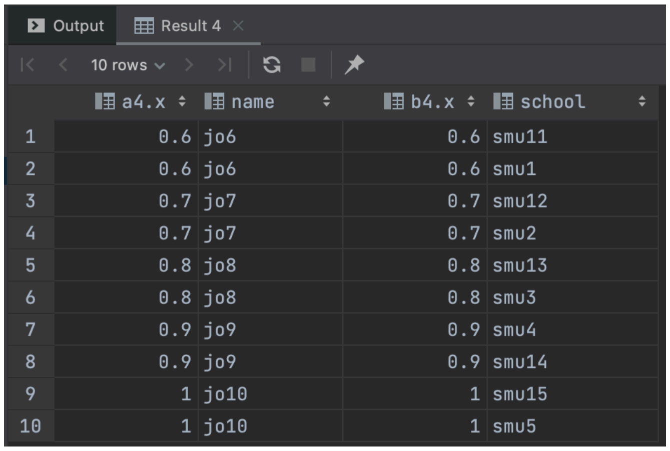


	x	name
1	0.1	jo1
2	0.2	jo2
3	0.3	jo3
4	0.4	jo4
5	0.5	jo5
6	0.6	jo6
7	0.7	jo7
8	0.8	jo8
9	0.9	jo9
10	1	jo10

x	school



1	1.3	smu18
2	1.4	smu9
3	1.3	smu8
4	1.4	smu19
5	1.1	smu16
6	0.7	smu2
7	0.6	smu11
8	1.2	smu7
9	0.7	smu12
10	0.9	smu4
11	1	smu5
12	0.8	smu3
13	1.2	smu17
14	0.8	smu13
15	1.1	smu6
16	0.9	smu14
17	1.5	smu10
18	0.6	smu1
19	1	smu15
20	1.5	smu20



	a4.x	name	b4.x	school
1	0.6	jo6	0.6	smu11
2	0.6	jo6	0.6	smu1
3	0.7	jo7	0.7	smu12
4	0.7	jo7	0.7	smu2
5	0.8	jo8	0.8	smu13
6	0.8	jo8	0.8	smu3
7	0.9	jo9	0.9	smu4
8	0.9	jo9	0.9	smu14
9	1	jo10	1	smu15
10	1	jo10	1	smu5

Therefore the lower bound is 0, the upperbound is 10000.

2. Consider the following schedule for four transactions:

Time	T1	T2	T3	T4
1			X3 = Read(X)	
2	Y1 = Read(Y)			
3				Z4 = Read(Z)
4		X2 = Read(X)		
5			X3 = X3 + 1	
6	Z1 = Read(Z)			
7		X2 = X2 * 2		
8	Z1 = Z1 + Y1			
9	Write(Z, Z1)			
10				Z4 = Z4 * 3
11			Y3 = Y3 – X3	
12			Write(X, Y3)	
13		Y2 = Read(Y)		
14		Y2 = Y2 + 4		
15		Write(Y, Y2)		
16				Write(Z, Z4)
17		Write(X,X2)		

How to read the table:

- X, Y, Z are data stored on the database in the disk
- X1..X4, Y1..Y4, Z1..Z4 are local variables stored in memory and only accessible via the corresponding transactions (T1 for X1, Y1, T2 for X2, Y2 etc.). You can assume all local variables are initialized to 0.
- The Read() operation read in the corresponding data on the disk and store it in the local variable
- All the operations (*, +, - etc.) only operate on the local variable, and do not affect the data on the disk
- The Write() command write the value of the local variable onto the corresponding data on the disk.

a. Is the above schedule serializable? If it is, list all possible equivalent serial schedule. If not, explain why not.

b. Now suppose we want to implement 2-phase locking on the transactions. Assume we use the following convention:

1. Every time a transaction wants to read/write an item in the disk for the first time, it will request a share/exclusive lock (respectively). We denote it as the operation S-Lock(), X-Lock() respectively
2. If a request is granted, the transaction can proceed immediately and execute the corresponding command
3. Otherwise, the transaction will use the wait-die policy to determine whether it will wait or abort.
4. If a transaction waits, it will wait until the lock is released and attempt to obtain it. It will try to obtain the lock immediately after it was released by another transaction.
5. If it can obtain the lock, it will execute all commands up to the time that the lock is requested/obtained.
6. If a transaction aborts, it will not be restarted
7. If there is more than one transaction waiting for the same lock, the lock is always granted to the transaction Ti with the smallest value of i.
8. We assume a transaction immediately commits and release all the locks after the last operation is

successfully executed

successfully executed.

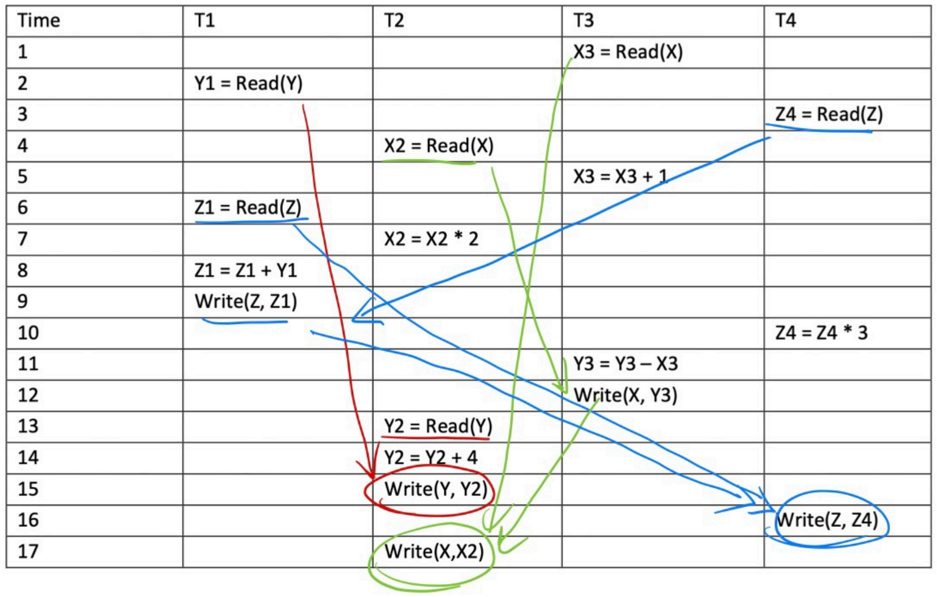
Below is a simple example

	Before adding lock command			After adding lock command		
	T1	T2	T3	T1	T2	T3
1	X1 = Read(X)			S-Lock(X); X1=Read(X)		
2		X2 = Read(X)			S-Lock(X); X2=Read(X)	
3			Y3 = Read(Y)			S-Lock(Y); Y3=Read(Y)
4			Y3 = Y3 + 1			Y3 = Y3 + 1
5			Write(Y, Y3)			X-Lock(Y); Write(Y, Y3)
6		Y2 = Read(Y)			S-Lock(Y); Wait	
7	Y1= Read(Y)			S-Lock(Y); Wait		
8	Y1 = Y1 + 1			---		
9			Z1 = Read(Z)			S-Lock(Z); Z1=Read(Z)
10			Z1=Z1+1			Z1=Z1+1
11			Write(Z, Z1)			X-Lock(Z); Write(Z, Z1); End T3,
				S-Lock(Y); Y1= Read(Y); Y1 = Y1 + 1		
12		X2 = Y2 + 1			----	
13	Write(Y, Y1)			X-Lock(Y); Write(Y, Y1)		

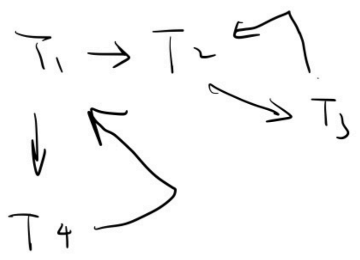
Apply the same notation to the 4 transactions on the first table, and show the execution in the table form as above.

Solution:

a. Because there is a circle, it is not schedule serializable.



Time	T1	T2	T3	T4
1			X3 = Read(X)	
2	Y1 = Read(Y)			
3				Z4 = Read(Z)
4		X2 = Read(X)		
5			X3 = X3 + 1	
6	Z1 = Read(Z)			
7		X2 = X2 * 2		
8	Z1 = Z1 + Y1			
9	Write(Z, Z1)			
10				Z4 = Z4 * 3
11			Y3 = Y3 – X3	
12			Write(X, Y3)	
13		Y2 = Read(Y)		
14		Y2 = Y2 + 4		
15		Write(Y, Y2)		
16				Write(Z, Z4)
17		Write(X,X2)		



$$\begin{tikzcd} T_1 \arrow[r] \arrow[d] & T_2 \arrow[dr] & \\ T_4 \arrow[u, bend right] & & T_3 \arrow[ul, bend right] \end{tikzcd}$$

It's has circle

∴ It's not serializable.

And whatever Transaction swap to the first, it always have the conflicts, it is not schedule seriaizable.

T1 first					T2 first				

Time	T1	T2	T3	T4	Time	T1	T2	T3	T4
1	Y1 = Read(Y)				1		X2 = Read(X)		
2	Z1 = Read(Z)				2		X2 = X2 * 2		
3	Z1 = Z1 + Y1				3		Y2 = Read(Y)		
4			X3 = Read(X)		4		Y2 = Y2 + 4		
5				Z4 = Read(Z)	5		Write(Y, Y2)		
6	Write(Z, Z1)				6			X3 = Read(X)	
7		X2 = Read(X)			7		Write(X, X2)		
8			X3 = X3 + 1		8	Y1 = Read(Y)			
9		X2 = X2 * 2			9				Z4 = Read(Z)
10				Z4 = Z4 * 3	10			X3 = X3 + 1	
11			Y3 = Y3 - X3		11	Z1 = Read(Z)			
12			Write(X, Y3)		12	Z1 = Z1 + Y1			
13		Y2 = Read(Y)			13	Write(Z, Z1)			
14		Y2 = Y2 + 4			14				Z4 = Z4 * 3
15		Write(Y, Y2)			15			Y3 = Y3 - X3	
16				Write(Z, Z4)	16			Write(X, Y3)	
17		Write(X, X2)			17				Write(Z, Z4)
T3 first					T4 first				
Time	T1	T2	T3	T4	Time	T1	T2	T3	T4
1			X3 = Read(X)		1				Z4 = Read(Z)
2			X3 = X3 + 1		2				Z4 = Z4 * 3
3			Y3 = Y3 - X3		3			X3 = Read(X)	
4	Y1 = Read(Y)				4	Y1 = Read(Y)			
5				Z4 = Read(Z)	5		X2 = Read(X)		
6		X2 = Read(X)			6			X3 = X3 + 1	
7			Write(X, Y3)		7	Z1 = Read(Z)			
8	Z1 = Read(Z)				8				Write(Z, Z4)
9		X2 = X2 * 2			9		X2 = X2 * 2		
10	Z1 = Z1 + Y1				10	Z1 = Z1 + Y1			
11	Write(Z, Z1)				11	Write(Z, Z1)			
12				Z4 = Z4 * 3	12			Y3 = Y3 - X3	
13		Y2 = Read(Y)			13			Write(X, Y3)	
14		Y2 = Y2 + 4			14		Y2 = Read(Y)		
15		Write(Y, Y2)			15		Y2 = Y2 + 4		
16				Write(Z, Z4)	16		Write(Y, Y2)		
17		Write(X, X2)			17		Write(X, X2)		

b.

	Before adding lock command				After adding lock command			
Time	T1	T2	T3	T4	T1	T2	T3	T4
1			X3 = Read(X)				S-Lock(X); X3 = Read(X)	
2	Y1 = Read(Y)				S-Lock(Y) Y1 = Read(Y)			
3				Z4 = Read(Z)				S-Lock(z) Z4 = Read(Z)
4		X2 = Read(X)				S-Lock(X) X2 = Read(X)		
5			X3 = X3 + 1				X3 = X3 + 1	
6	Z1 = Read(Z)				S-Lock(Z) Z1 = Read(Z)			
7		X2 = X2 * 2				X2 = X2 * 2		
8	Z1 = Z1 + Y1				Z1 = Z1 + Y1			
9	Write(Z, Z1)				X-Lock(Z) Wait			
10				Z4 = Z4 * 3				Z4 = Z4 * 3
11			Y3 = Y3 - X3				Y3 = Y3 - X3	
12			Write(X, Y3)				X-Lock(X) Abort	
13		Y2 = Read(Y)				S-Lock(Y) Y2 = Read(Y)		
14		Y2 = Y2 + 4				Y2 = Y2 + 4		
15		Write(Y, Y2)				X-Lock(Y) Abort		
16				Write(Z, Z4)				X-Lock(Z): Wait T1 and T4 wait same lock, T1<T4 Abort

17		Write(X, X2)			X-Lock(Z):
					Write(Z, Z1);
					End T1;			