Name: Xiang Gu
EID: xg2847

Part 1:

	Load Time (random insert)	Bandwidth
Run locally	207s	6.18MB/s
Run on AWS	1551s	0.82MB/s

Part 2:

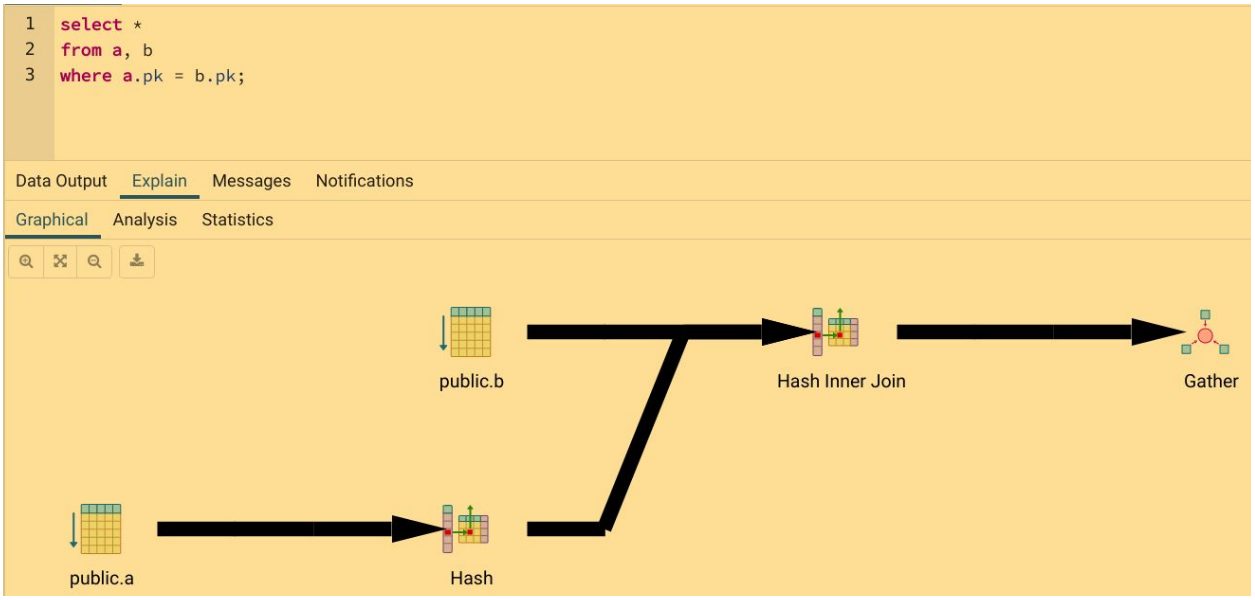
1. Comparison of our database instance with the one in the paper “Benchmarking Database Systems, A Systematic Approach”:

- There are four “basic” tables in the paper while we have only one;
- Each of the four tables contain 1000, 2000, 5000, 10000 tuples respectively, while our table contains 5,000,000 tuples;
- Each tuple in the four tables in the paper is 182 bytes long, while each tuple in our table is (about) 256 bytes.

2. How big a database (total rows and bytes) are you being asked to manage now?
There are 5,000,000 rows in each table and we have 6 copies of the table so we have in total 30,000,000 rows to manage, which is about 7.5 GB.

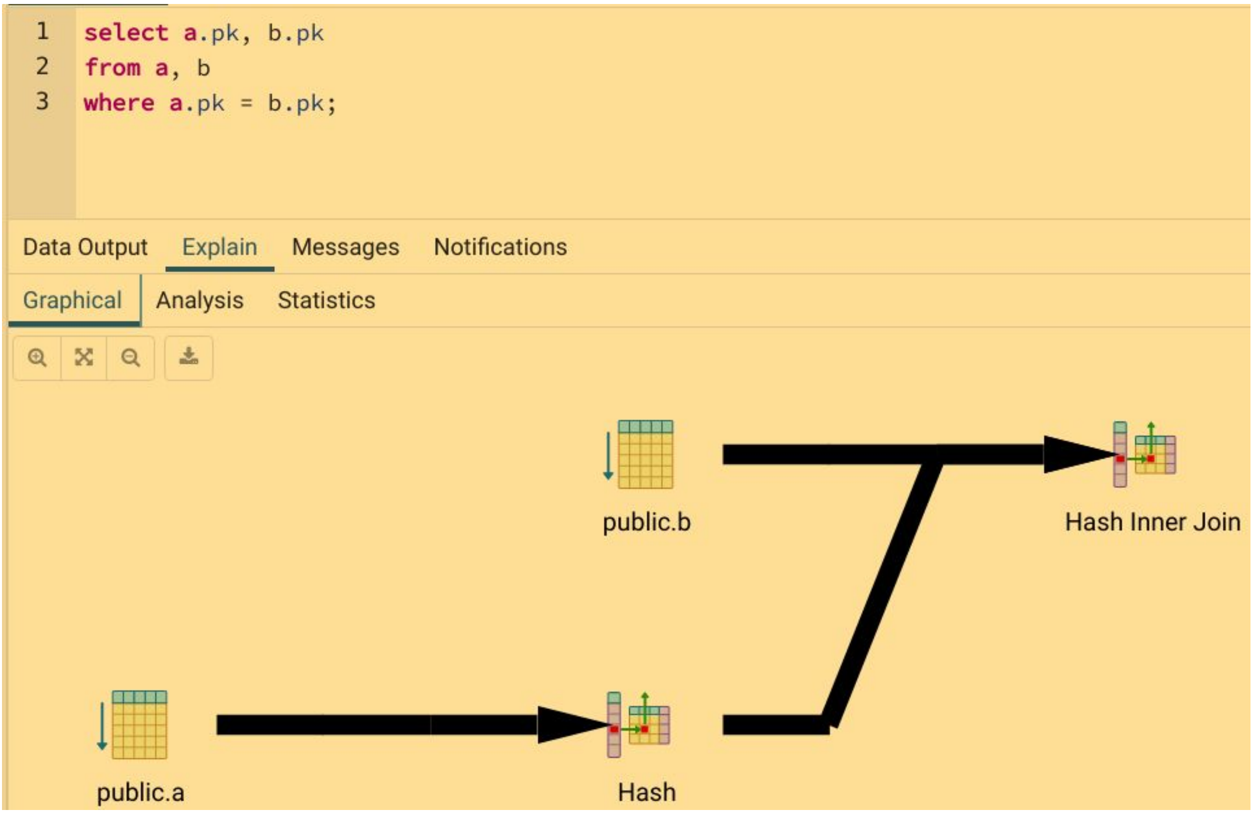
(A). No, my system chose the hash inner join algorithm, following sequential scans.

1.



```
select *
from a, b
where a.pk = b.pk;
```

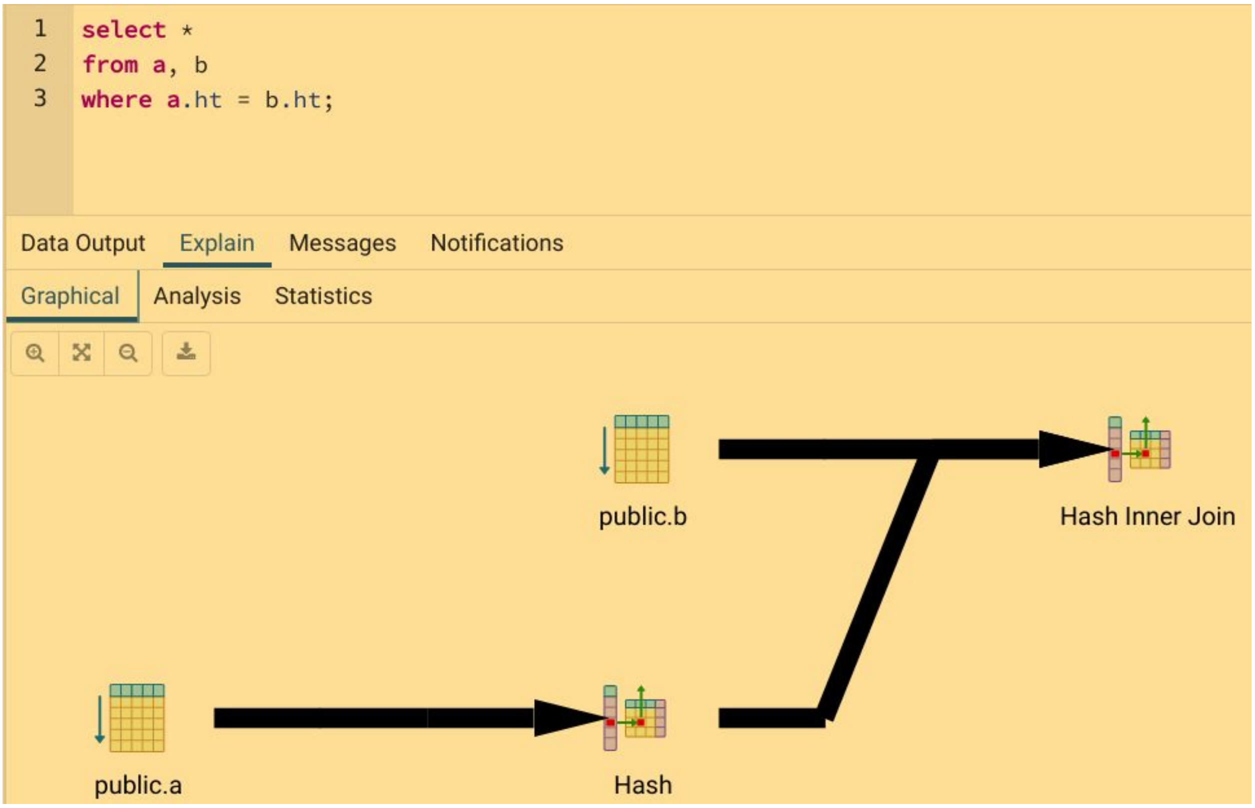
The additional query indeed does the push-down action on projection before joining (unfortunately, it’s not shown in the graphical explanation, but I have checked the analysis and indeed each row now has width = 4)



```
select a.pk, b.pk
from a, b
where a.pk = b.pk;
```

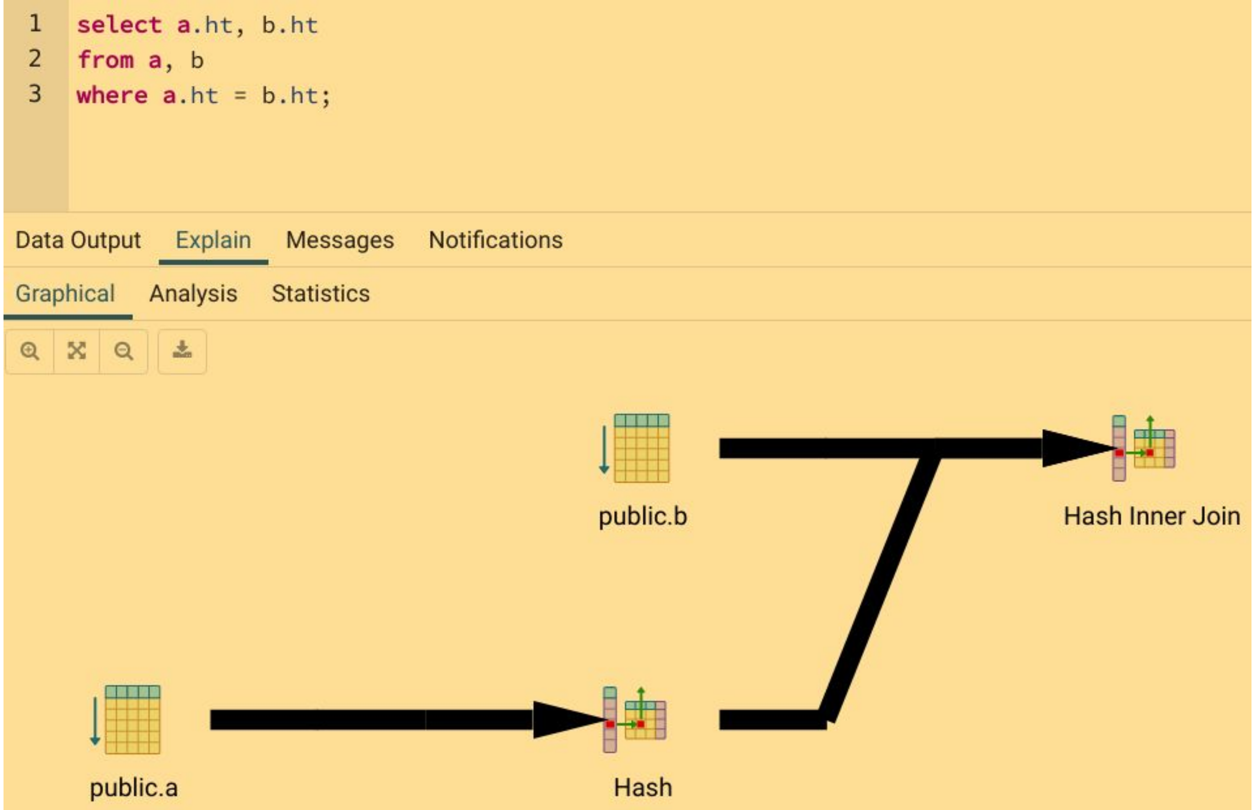
(B).Again, the hash inner join algorithm is chosen.

2.



```
select *
from a, b
where a.ht = b.ht;
```

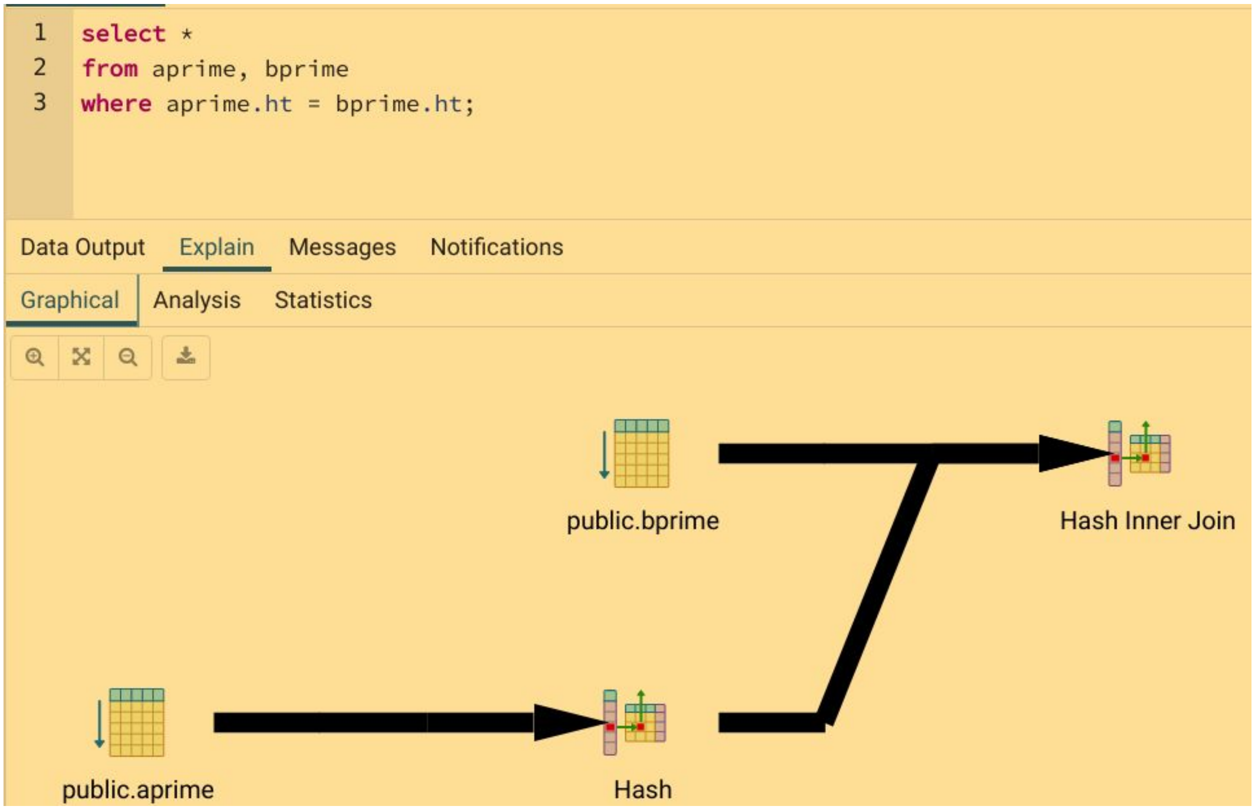
For the additional query, same join algorithm is chosen (hash inner join) but it does do the projection first before the join:



```
select a.ht, b.ht
from a, b
where a.ht = b.ht;
```

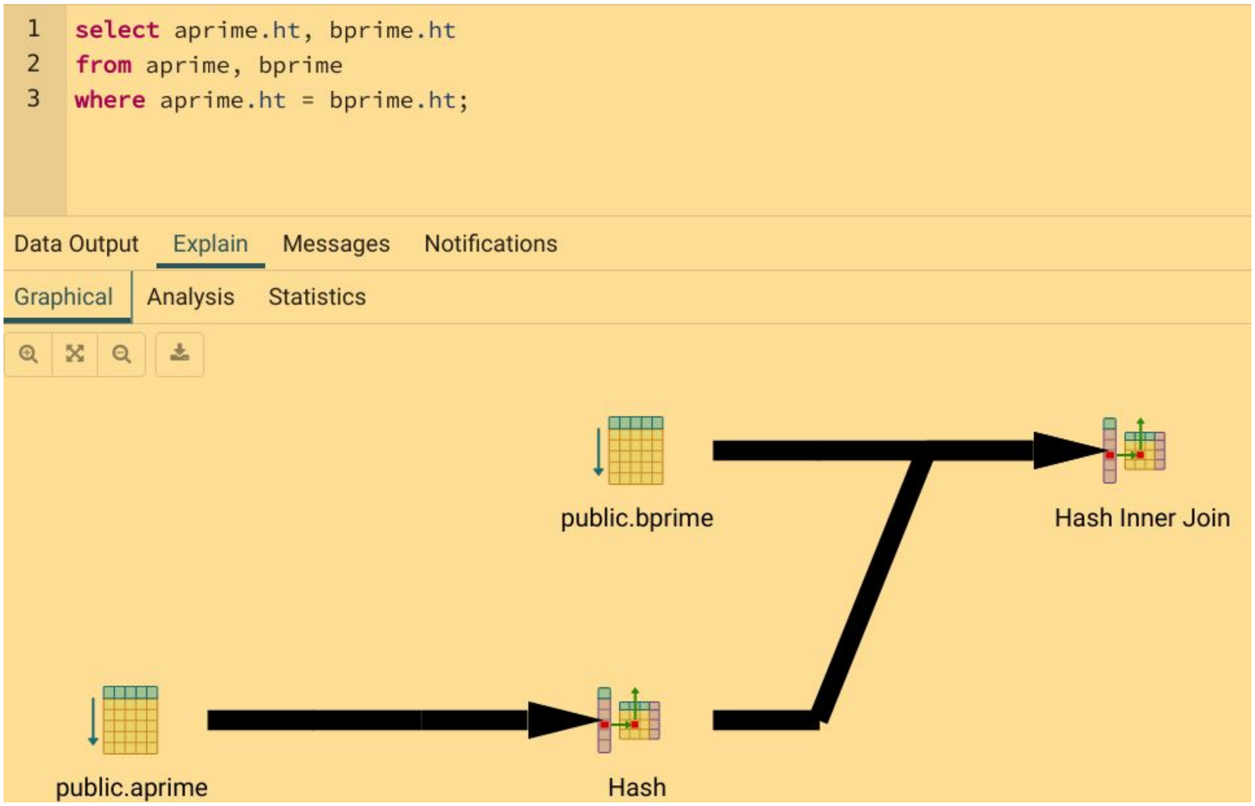
(C). No impact. Still, the hash inner join following sequential scans is used.

3.



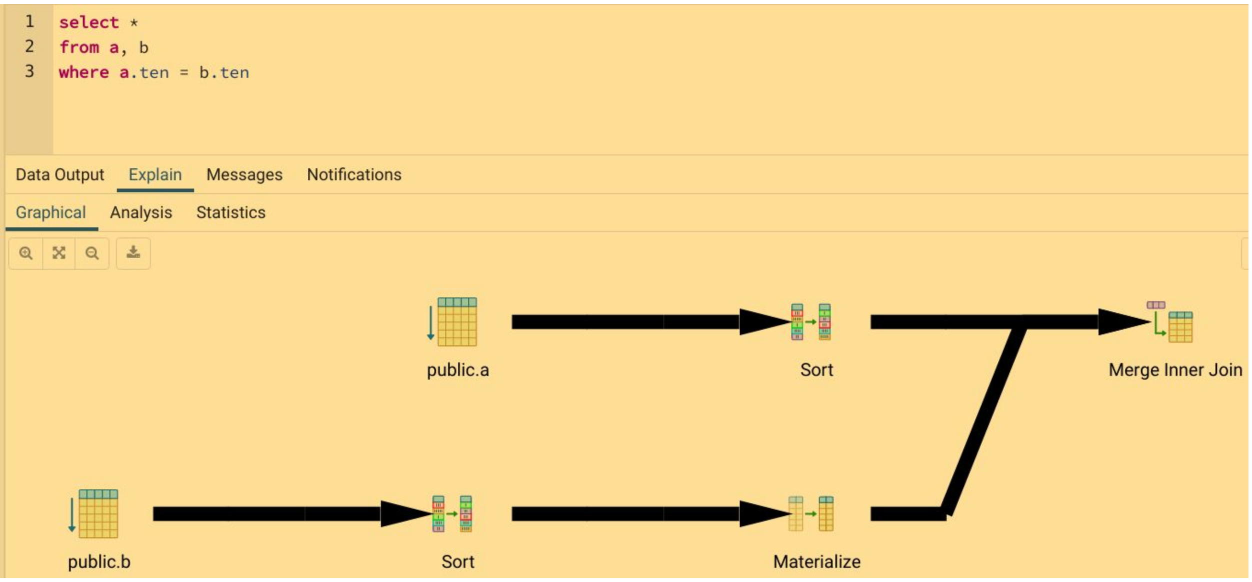
```
select *
from aprime, bprime
where aprime.ht = bprime.ht;
```

For the additional query, same thing happened again -- projection is pushed down before the join and hash inner join is used:



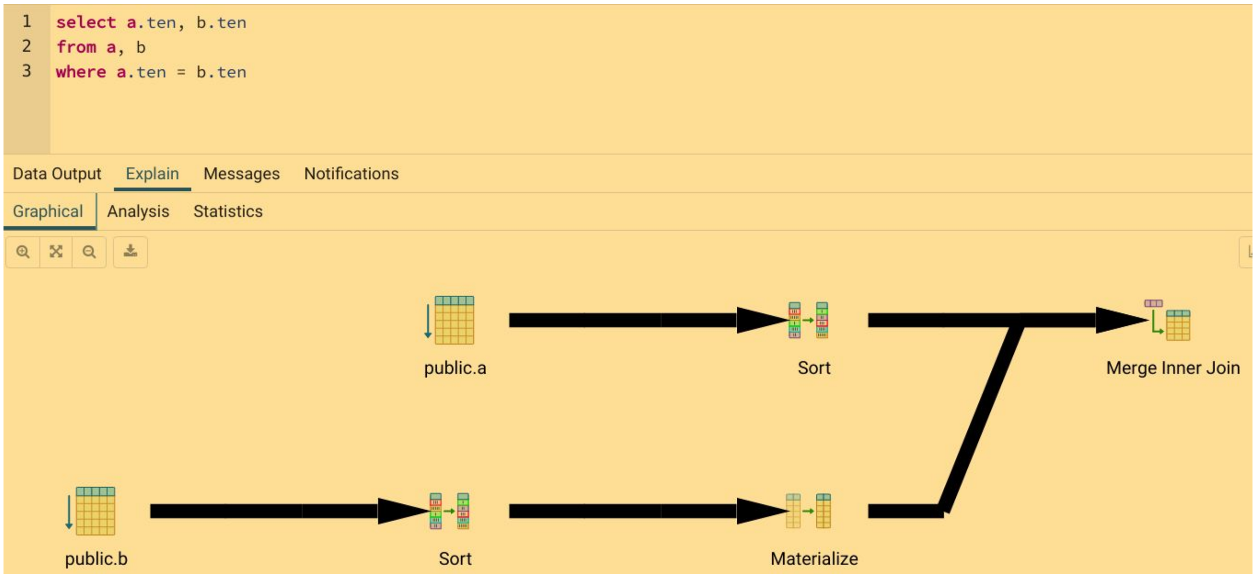
```
select aprime.ht, bprime.ht
from aprime, bprime
where aprime.ht = bprime.ht;
```

(D). Surprisingly, the merge inner join algorithm is chosen.
4.



```
select *
from a, b
where a.ten = b.ten
```

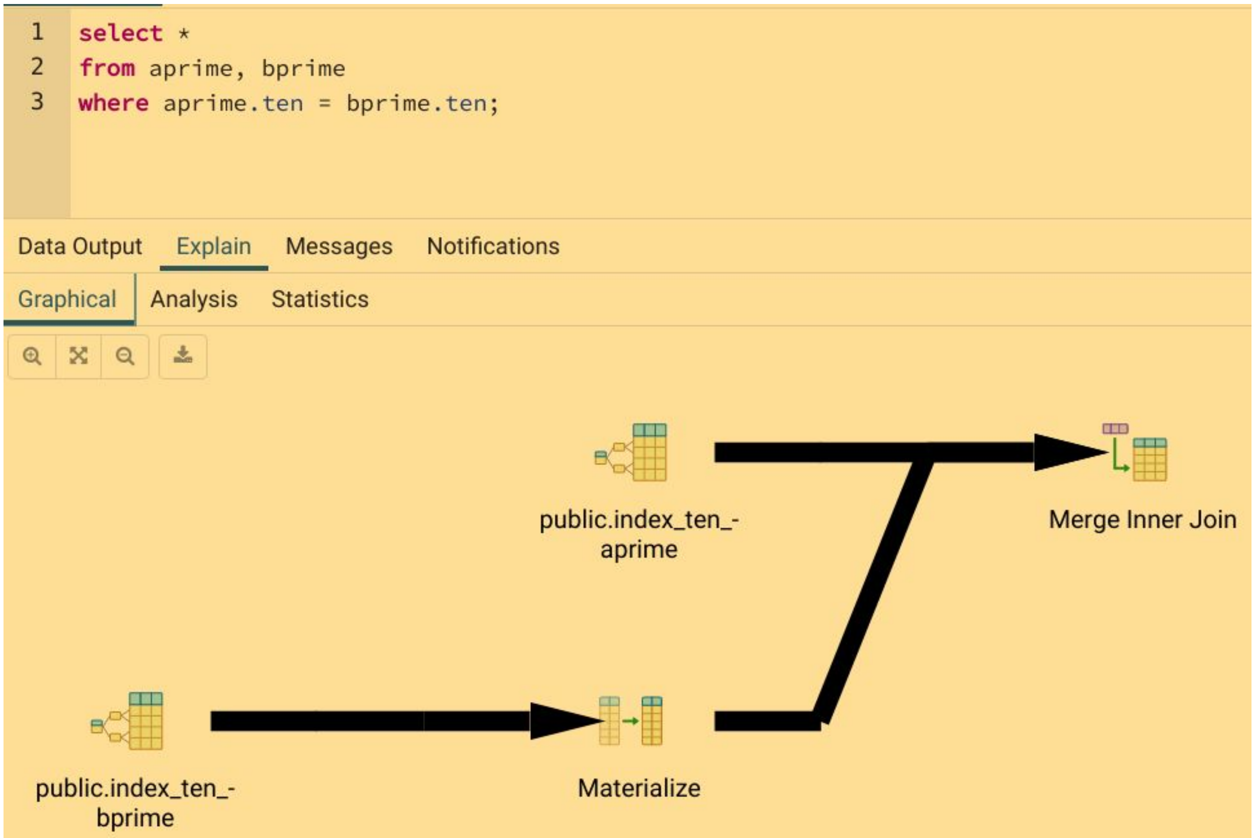
For the additional query, the same join algorithm (merge join) is chosen:



```
select a.ten, b.ten
from a, b
where a.ten = b.ten
```

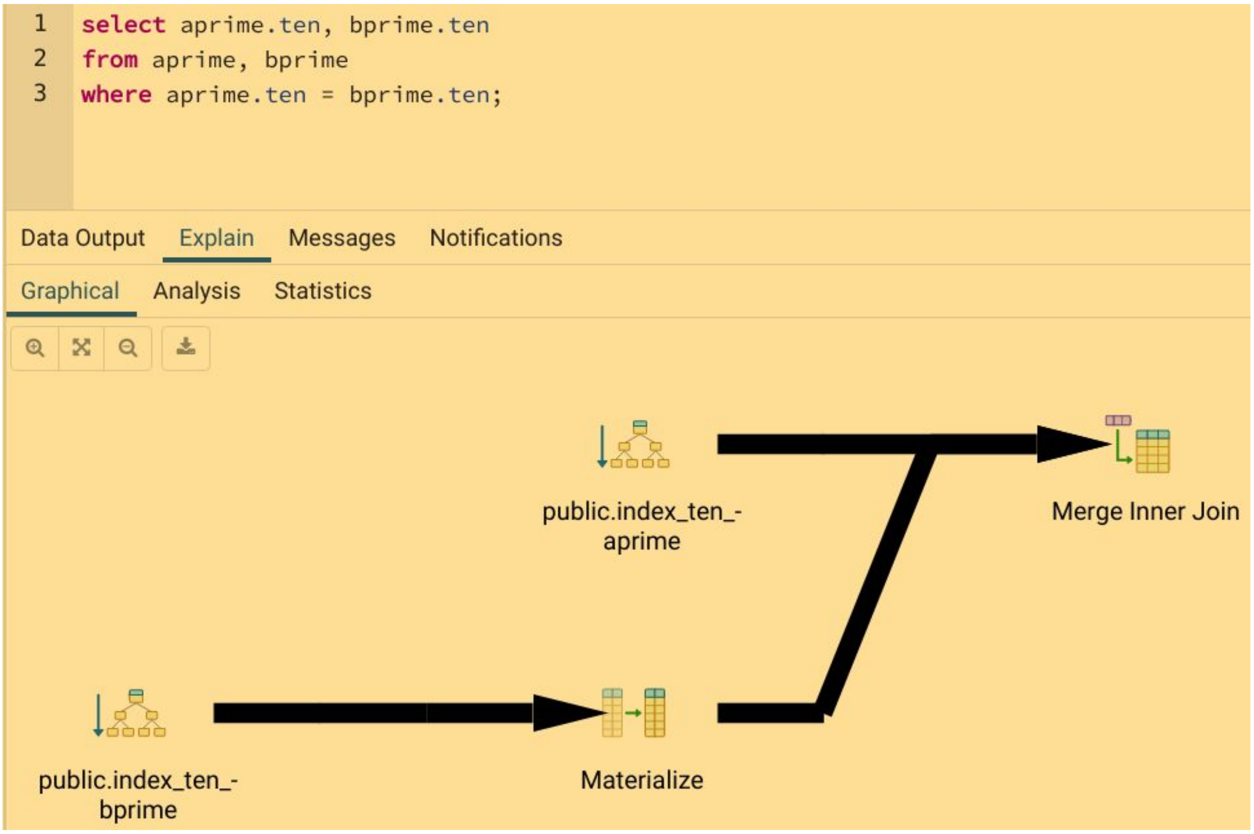
(E). Merge inner join algorithm is chosen, following the index scan on the index built on the 'ten' column.

5.



```
select *
from aprime, bprime
where aprime.ten = bprime.ten;
```

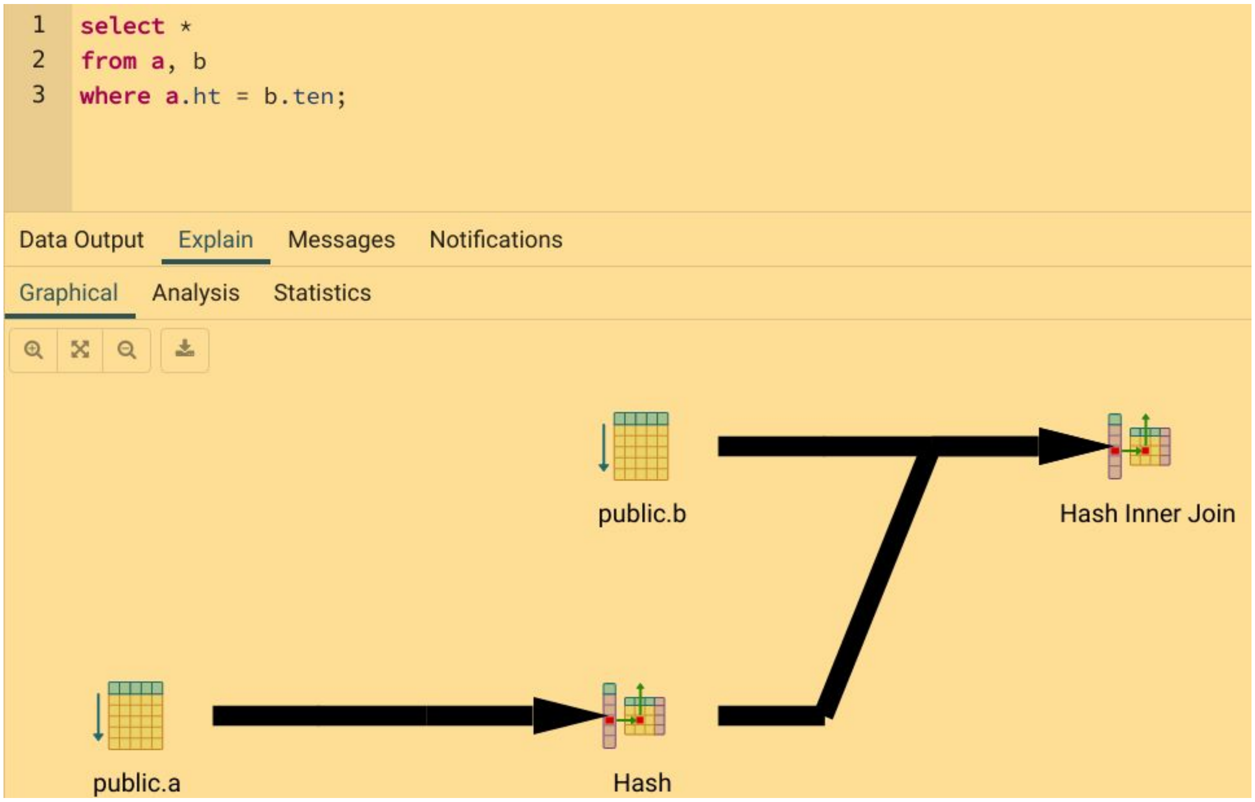
For the additional query, the merge inner join, following a index only scan, is chosen:



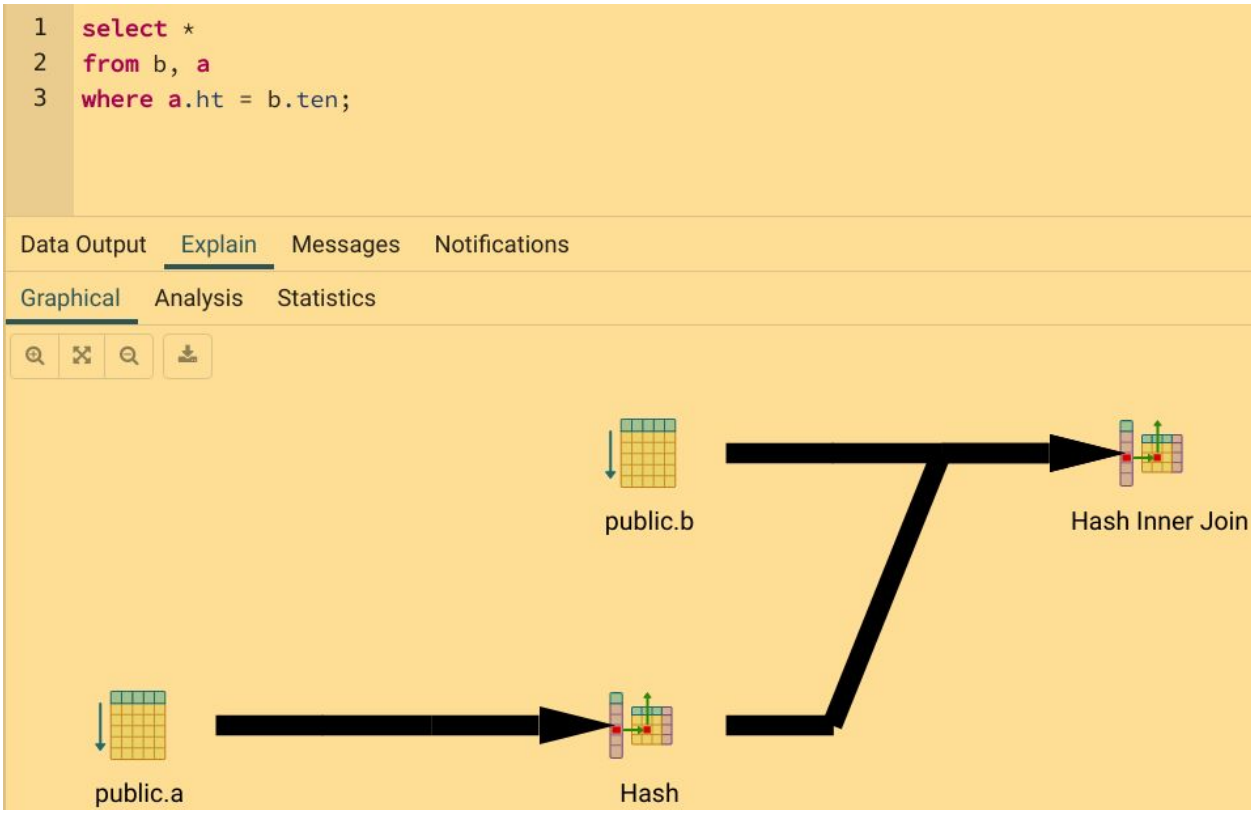
```sql
select aprime.ten, bprime.ten
from aprime, bprime
where aprime.ten = bprime.ten;
```

(F). Yes, the optimizer chose its own order for the join, independent of what order I wrote the query.

6. two different orderings have been tried:

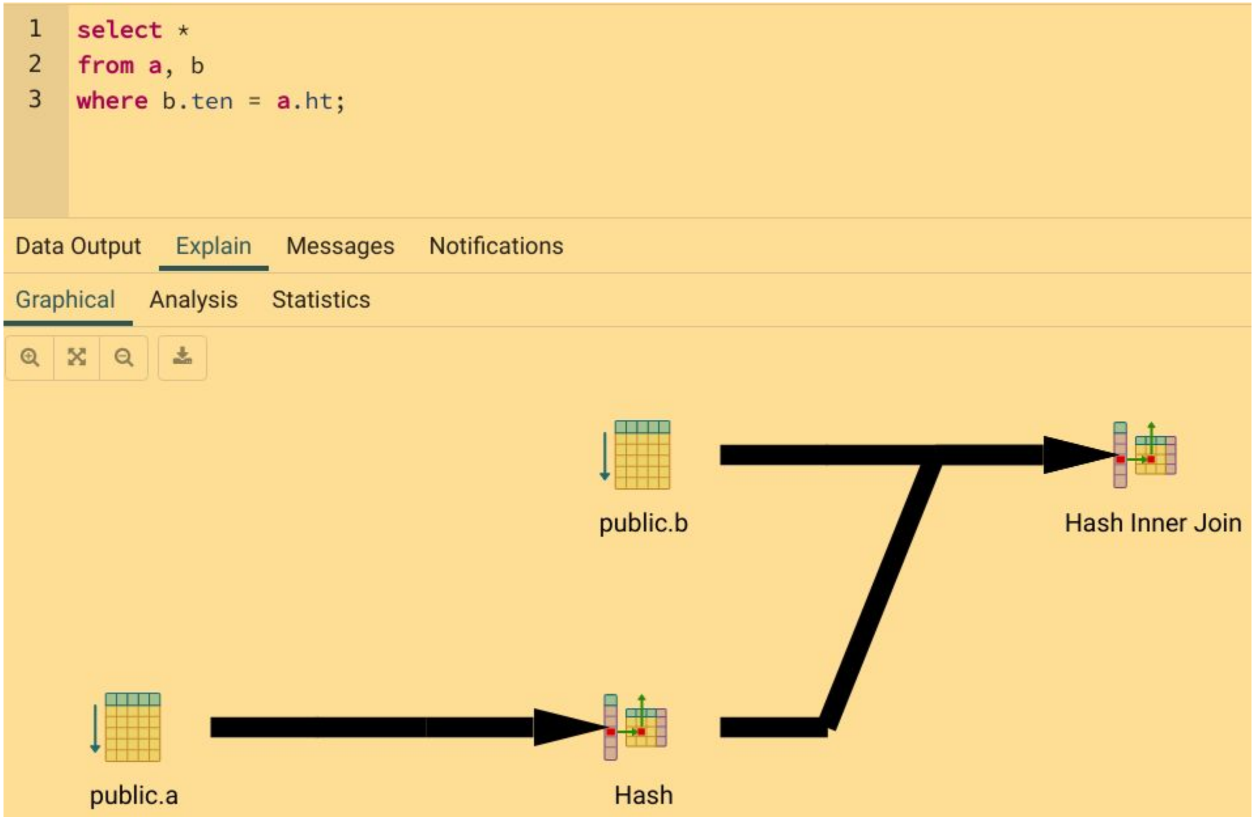


```
select *
from a, b
where a.ht = b.ten;
```

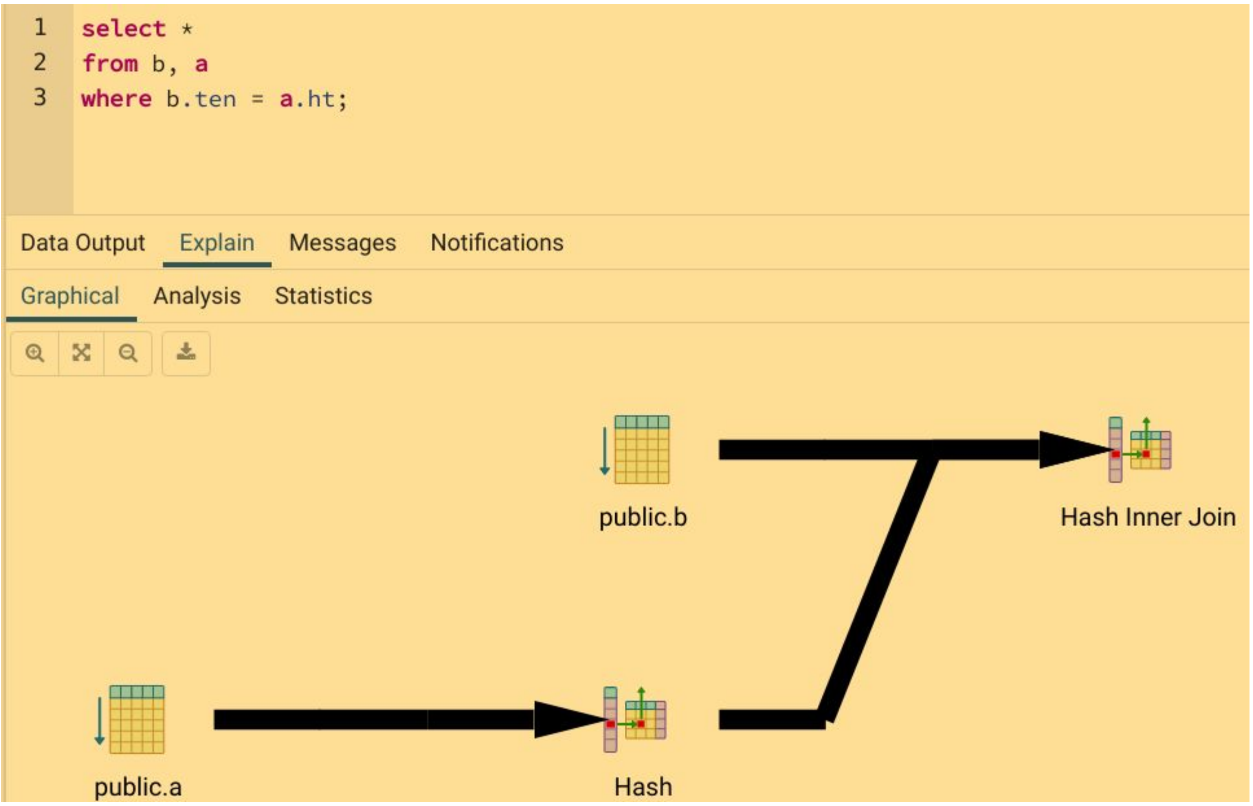


```
select *
from b, a
where a.ht = b.ten;
```

7. Again, same two orderings have been tried:



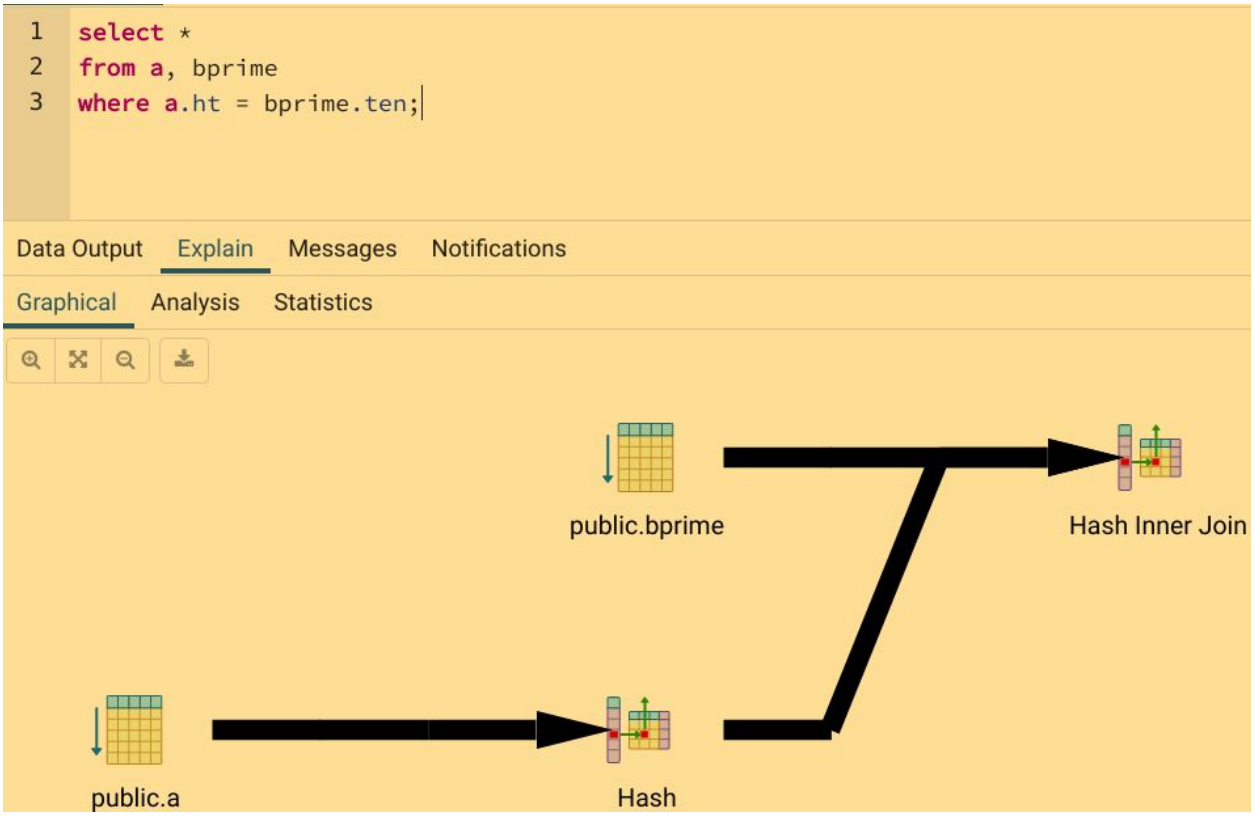
```
select *
from a, b
where b.ten = a.ht;
```



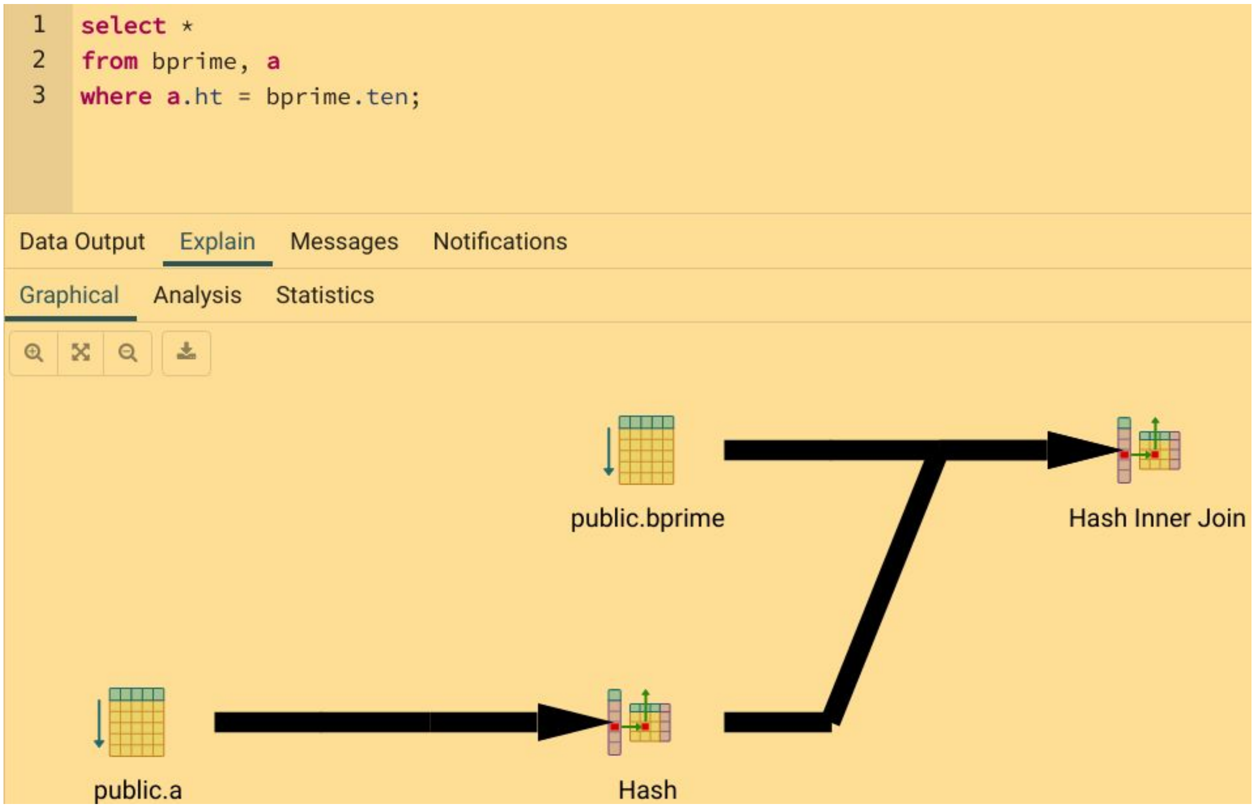
```
select *
from b, a
where b.ten = a.ht;
```

(G). The answer remains the same, that is, the optimizer chose its own join order, independent of what order I wrote the query.

8.

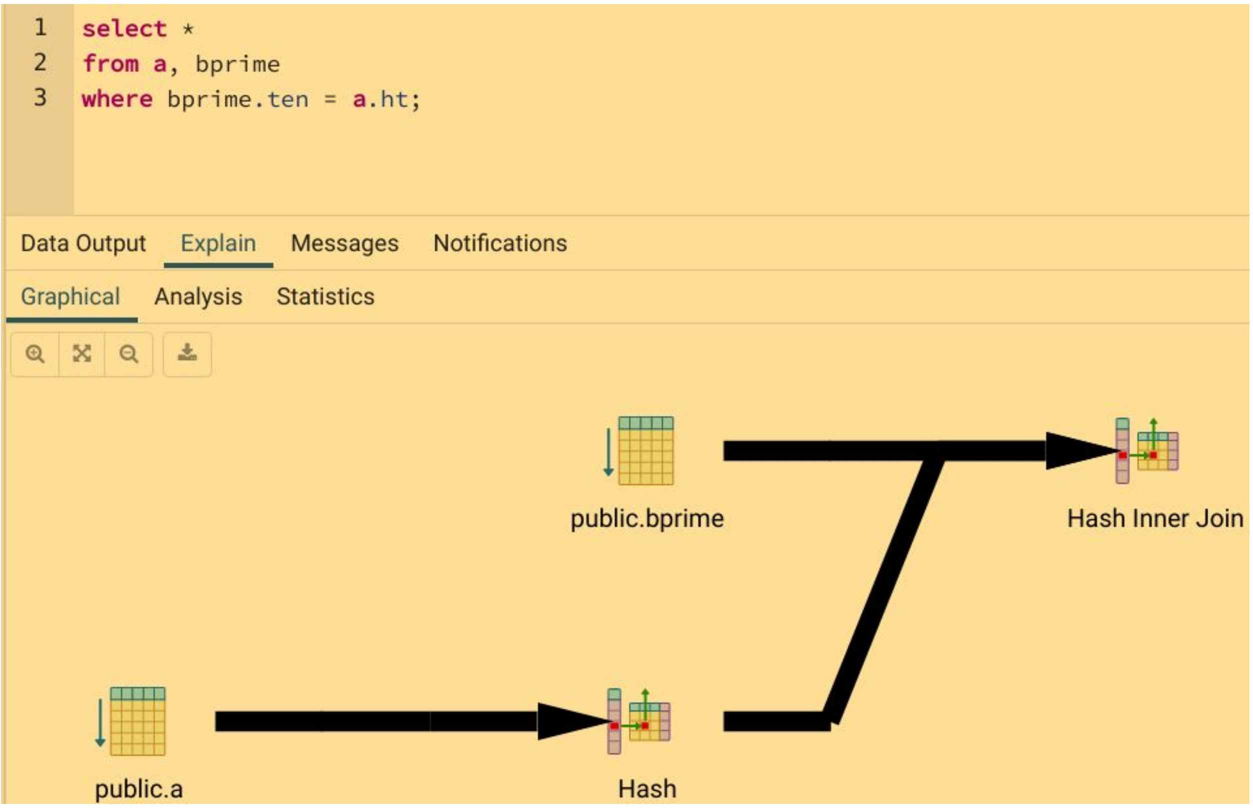


```
select *
from a, bprime
where a.ht = bprime.ten;
```

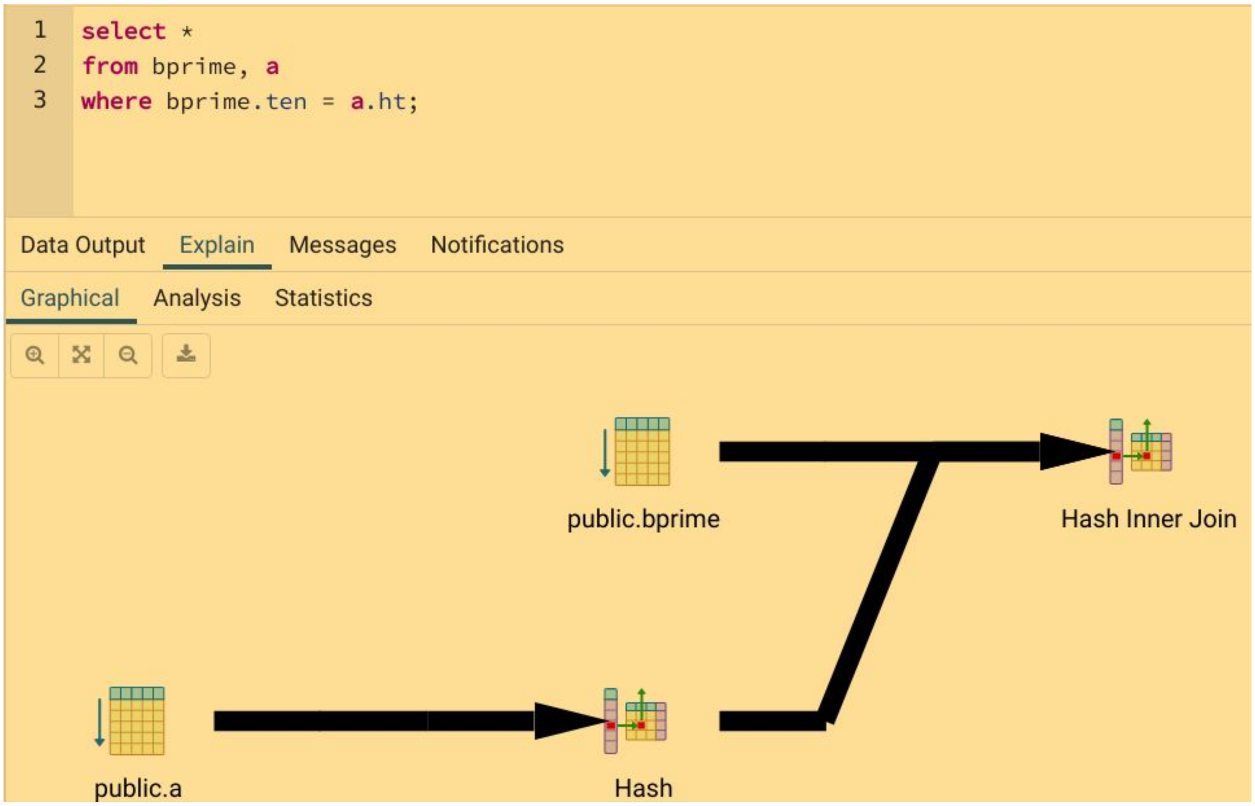


```
select *
from bprime, a
where a.ht = bprime.ten;
```

9.

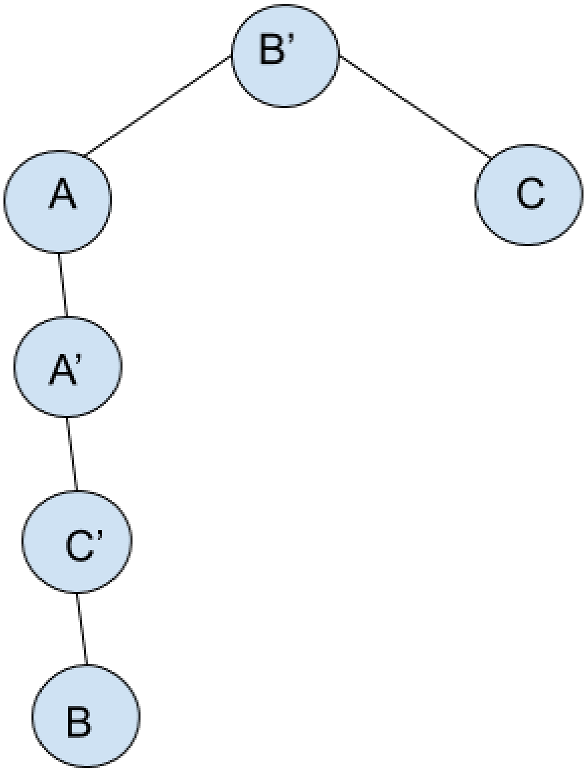


```
select *
from a, bprime
where bprime.ten = a.ht;
```



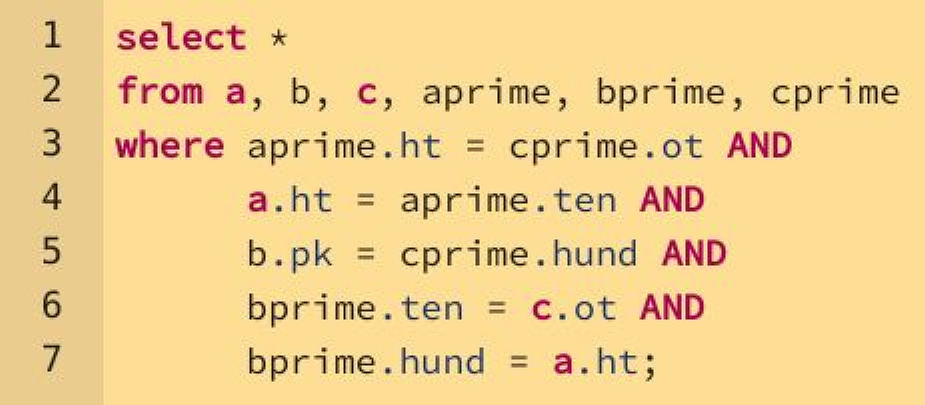
```
select *
from bprime, a
where bprime.ten = a.ht;
```

(H). Query graph of query 23 looks like:

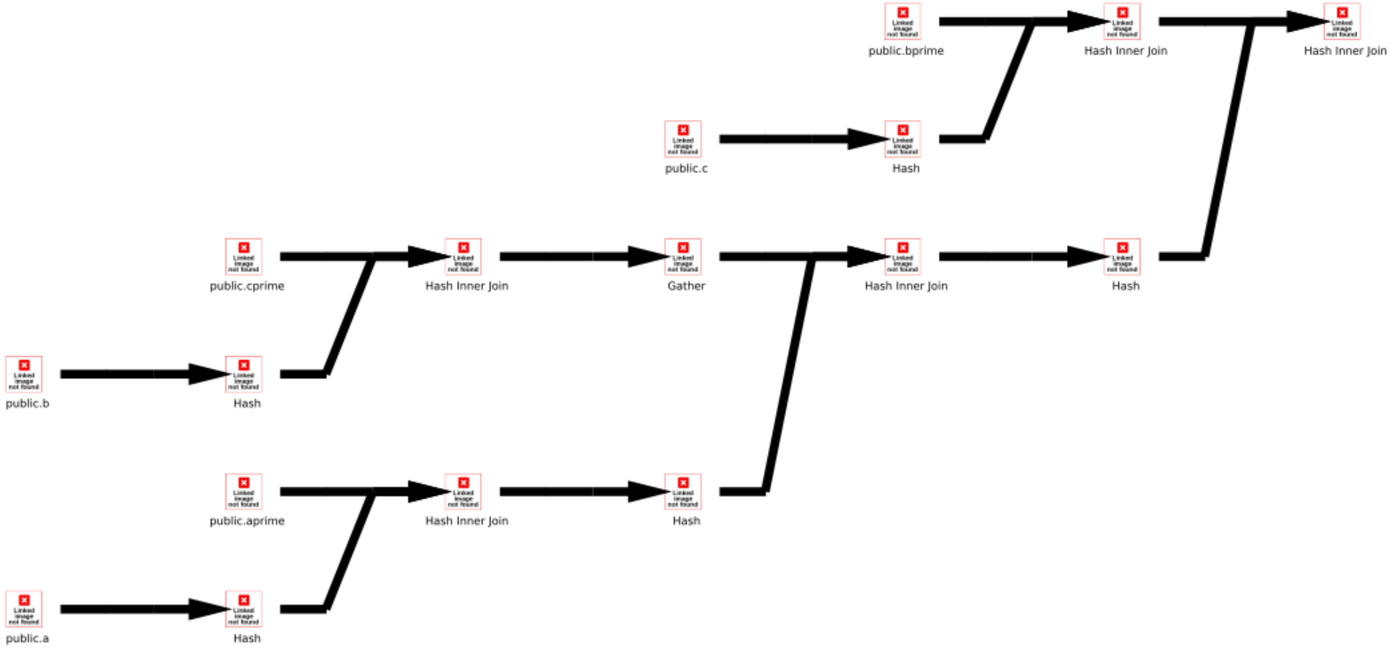


(I) The estimated number of rows of the resulting relation is ???

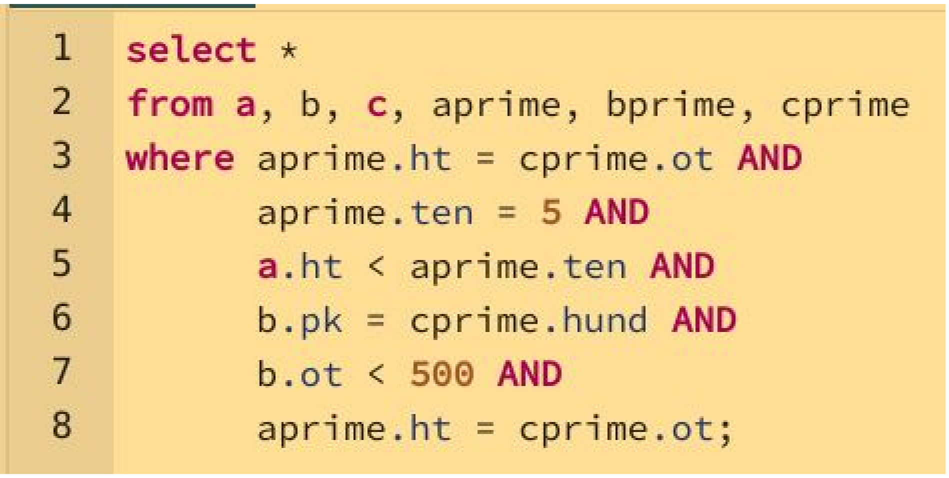
23.



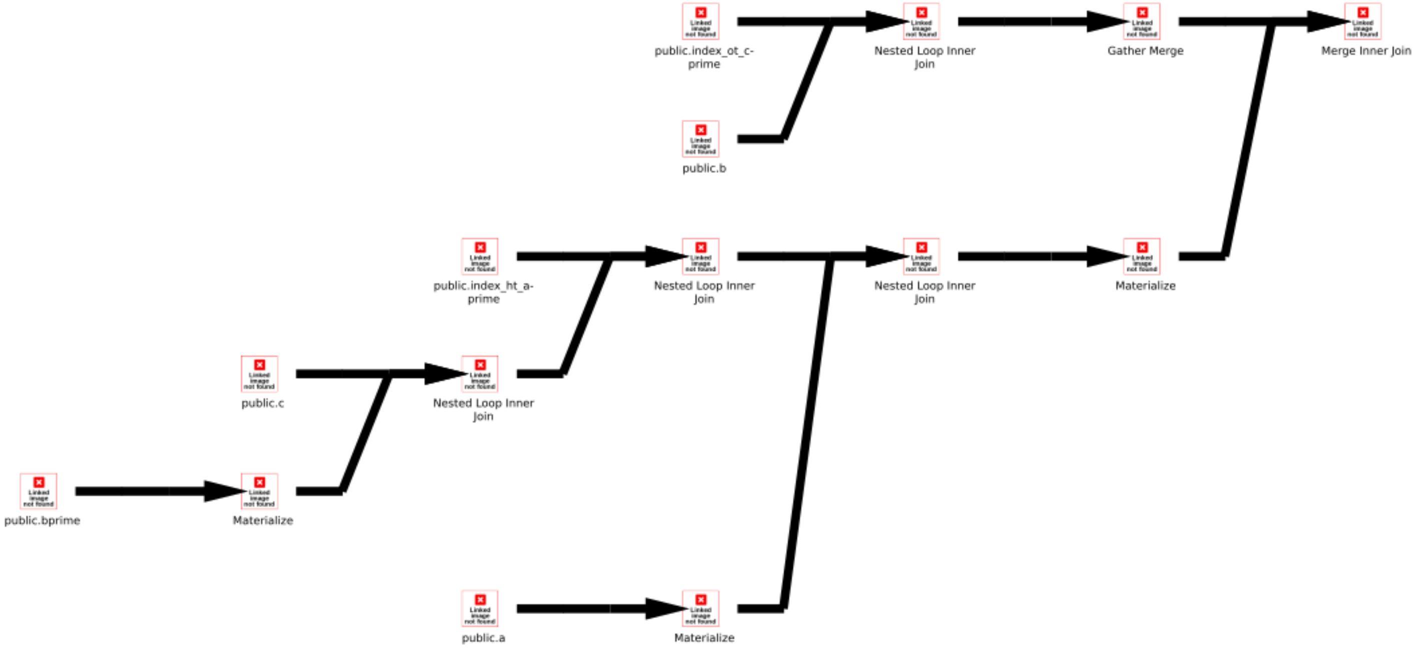
```
select *
from a, b, c, aprime, bprime, cprime
where aprime.ht = cprime.ot AND
      a.ht = aprime.ten AND
      b.pk = cprime.hund AND
      bprime.ten = c.ot AND
      bprime.hund = a.ht;
```



24.



```
select *
from a, b, c, aprime, bprime, cprime
where aprime.ht = cprime.ot AND
      aprime.ten = 5 AND
      a.ht < aprime.ten AND
      b.pk = cprime.hund AND
      b.ot < 500 AND
      aprime.ht = cprime.ot;
```


public.index_ot_c-prime
Nested Loop Inner Join
Gather Merge
Merge Inner Join
public.b
public.index_ht_a-prime
Nested Loop Inner Join
Nested Loop Inner Join
Materialize
public.c
Nested Loop Inner Join
public.bprime
Materialize
public.a
Materialize