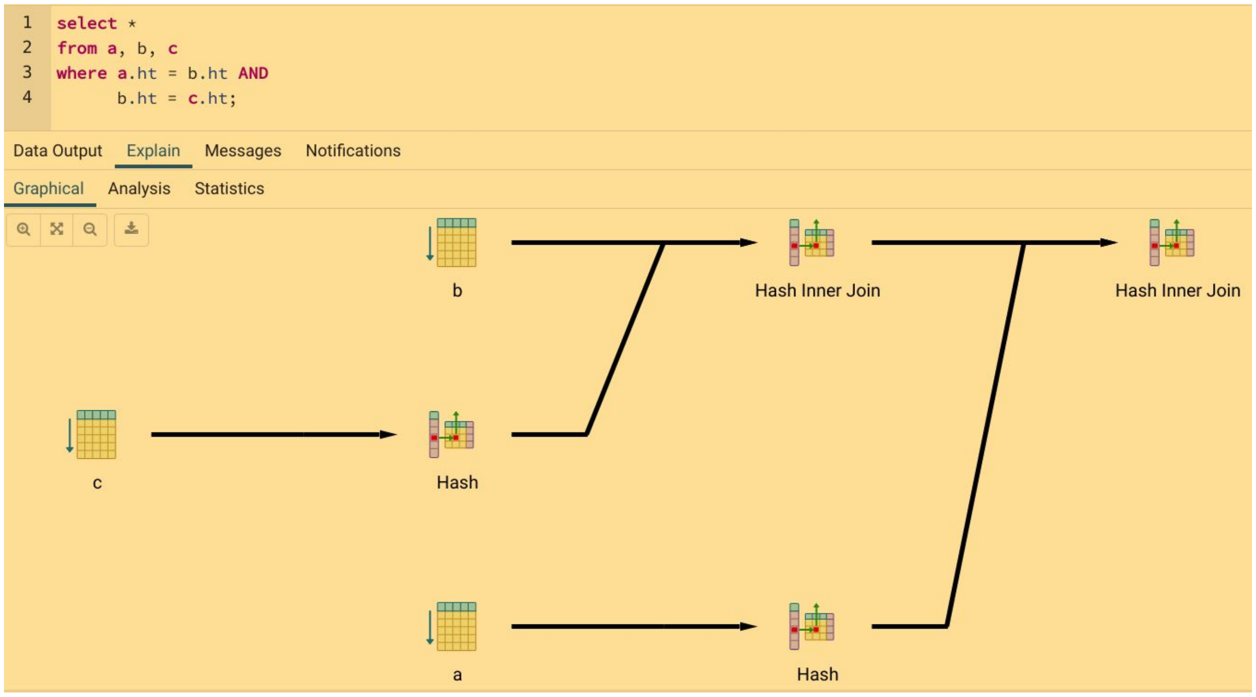
Part 1:

Client Location	Server Locaiton	Load Time	Bytes/second loaded
Local machine	Local machine	207s	6.18MB/s
Local machine	AWS	1551s	0.82MB/s
AWS	AWS	467.5s	2.74MB/s

Part 2:

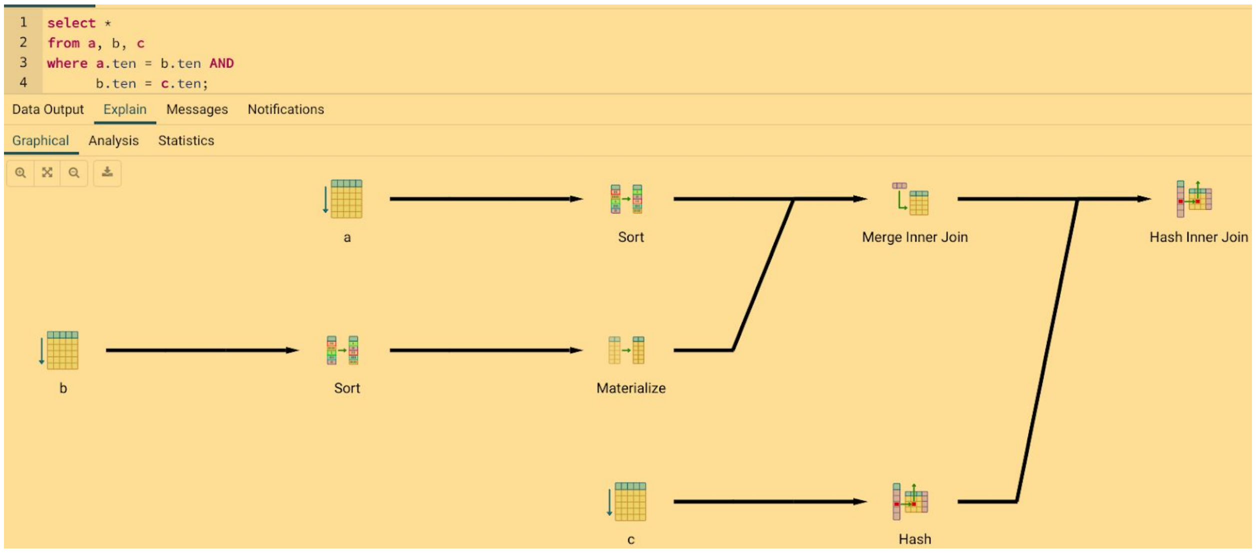
10.



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

The hash-join algorithm is chosen.

11.

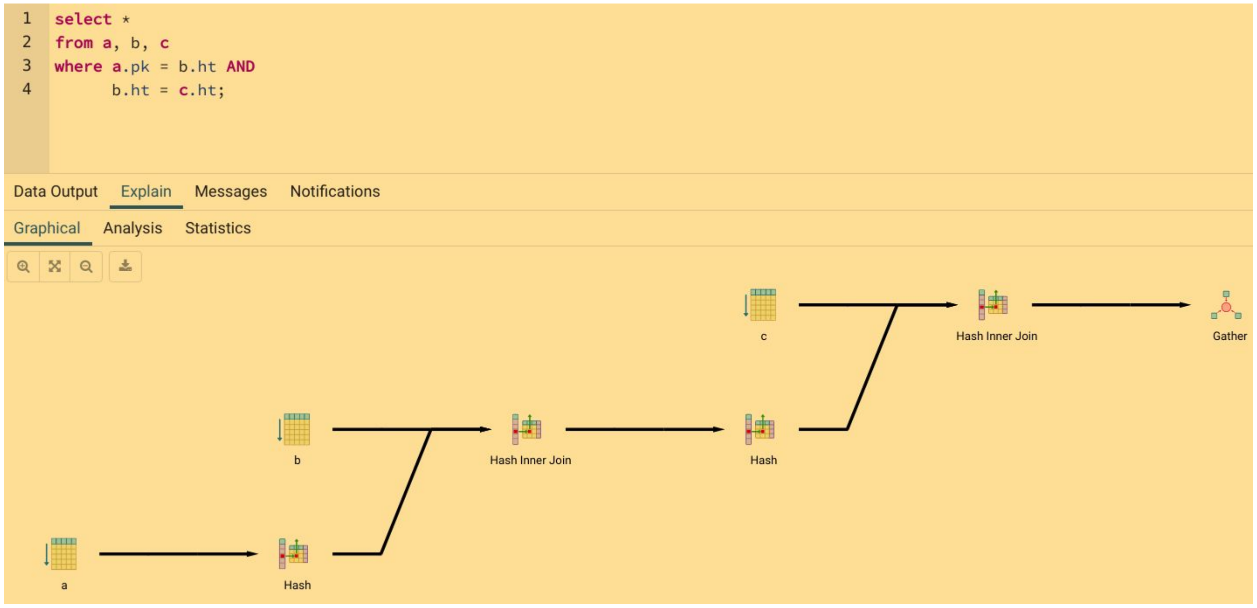


```sql
select *
from a, b, c
where a.ten = b.ten AND
      b.ten = c.ten;
```

Surprisingly, when we join on the “ten” column which has a higher selectivity, sort and merge join is selected first to join a and b. Then the result is hash-joined with the third table c.

12.
When joining on “ten” or “hund” column, it’s using the query plan in 11;
When joining on “ot”, “tt”, or “ht” column, it’s using the query plan in 10.

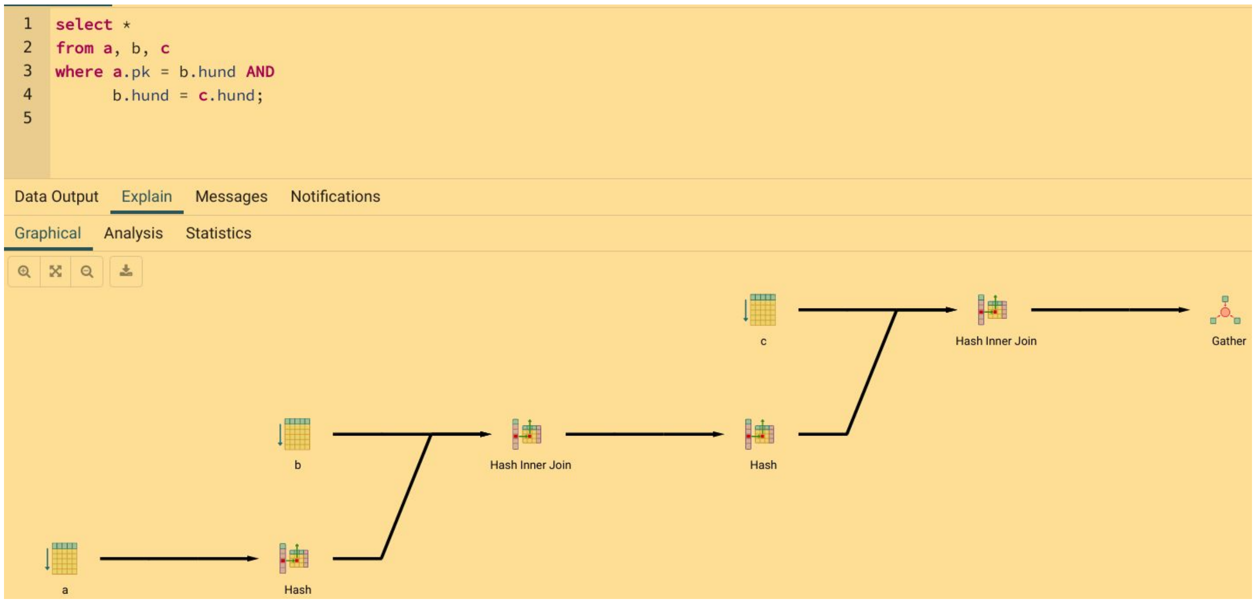
13.



```sql
select *
from a, b, c
where a.pk = b.ht AND
      b.ht = c.ht;
```

This time, again, hash join is chosen and used twice to get the final result.

14.

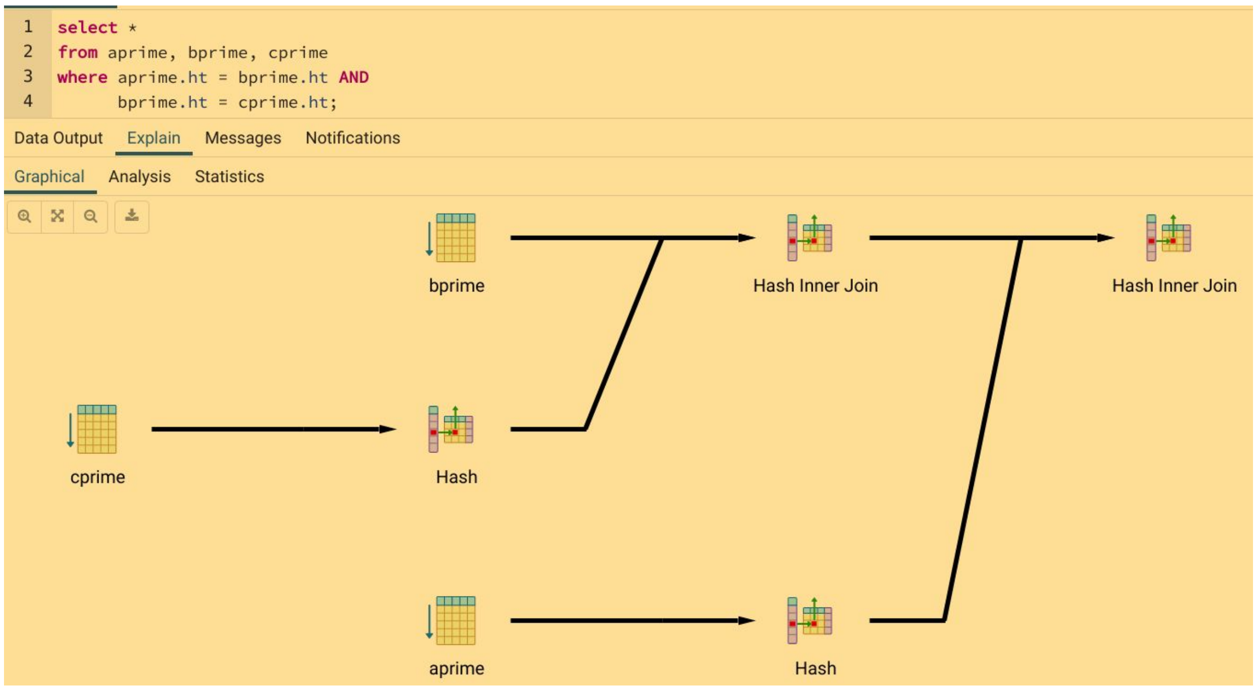


```
select *
from a, b, c
where a.pk = b.hund AND
      b.hund = c.hund;
```

Similarly, hash join is used.

15.
I tried to A.pk = b.ten = c.ten and it gives the same query plan! In other words, 13 and 14 produce the same query plan. Moreover, it’s always going to be the same plan no matter which column we choose to join on b and c.

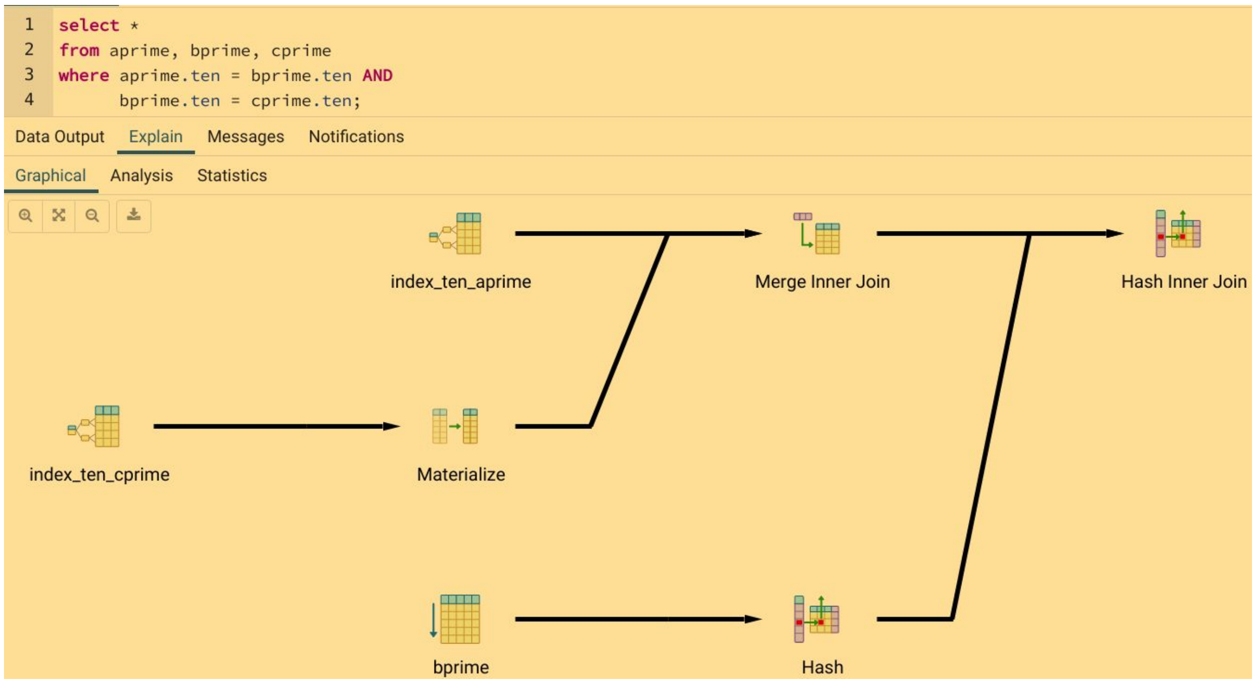
16.



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

Same, hash join is chosen.

17.

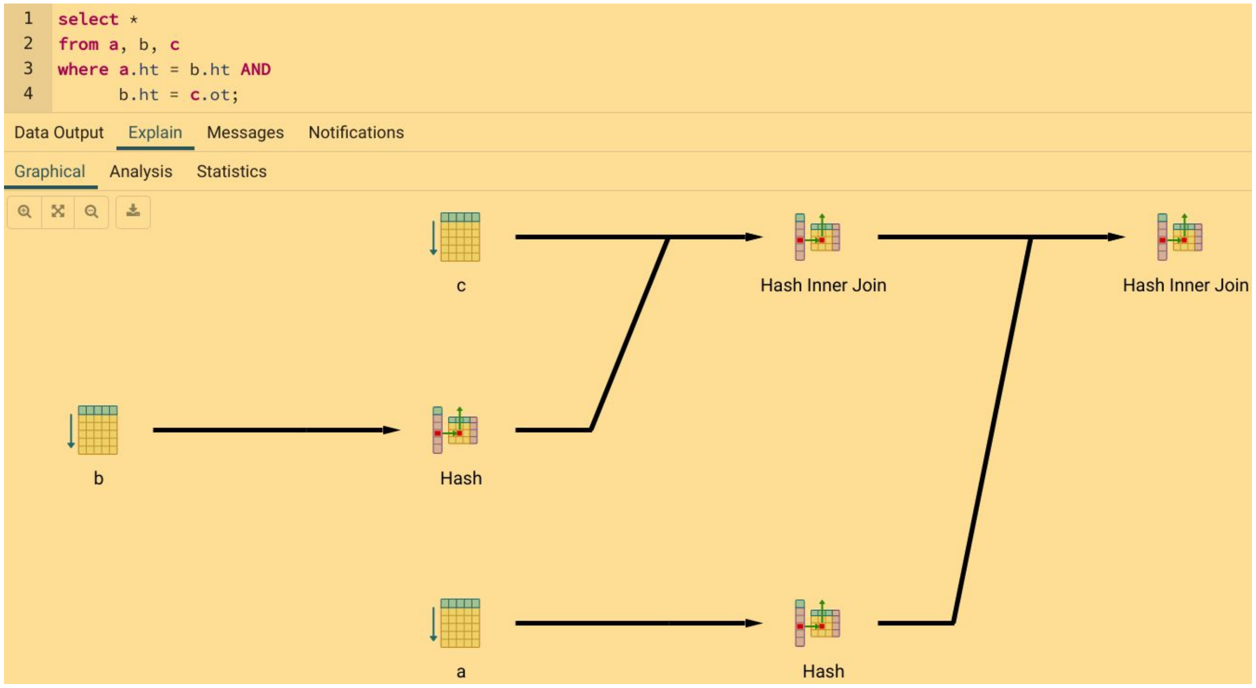


```sql
select *
from aprime, bprime, cprime
where aprime.ten = bprime.ten AND
      bprime.ten = cprime.ten;
```

Things are different now -- this time it will first use merge join, together with index scans, to join the first two relations. Then use hash join to join the resulting table with the third table.

18.
When joining on the “ten” or “hund” column, query plan 17 is chosen;
When joining on the “ot”, “tt” or “ht” column, query plan 16 is chosen.

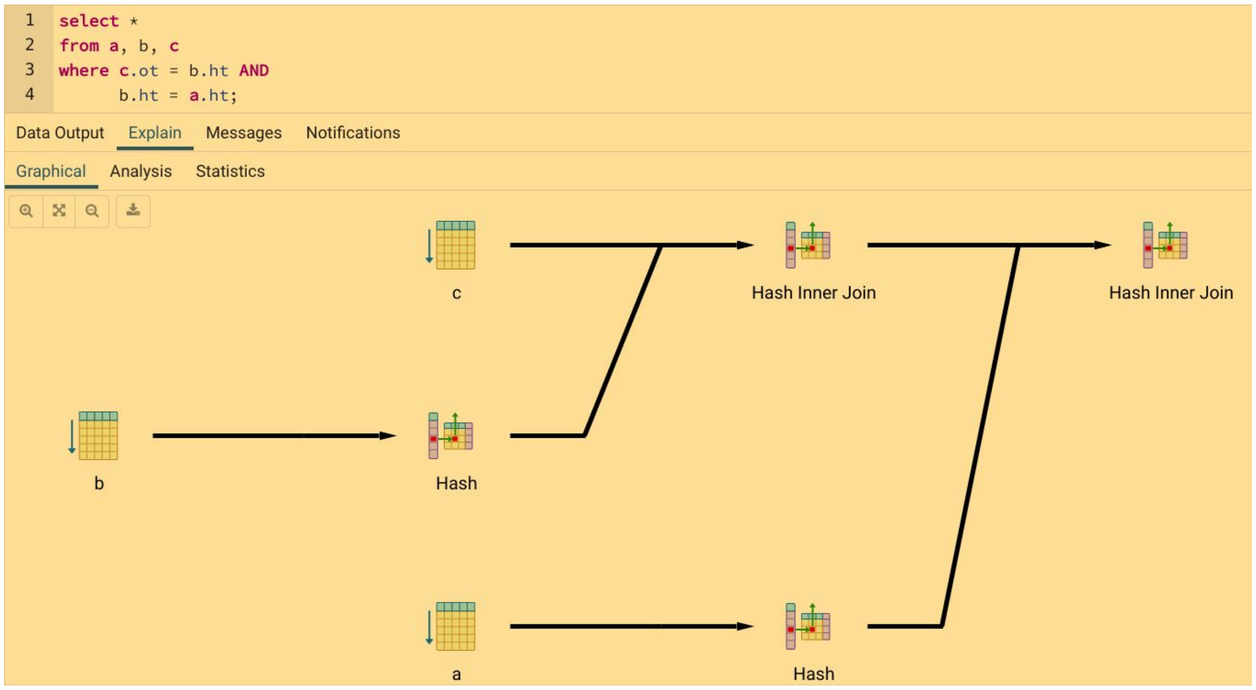
19.



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

Again, our most popular method -- hash join -- is chosen.

20.

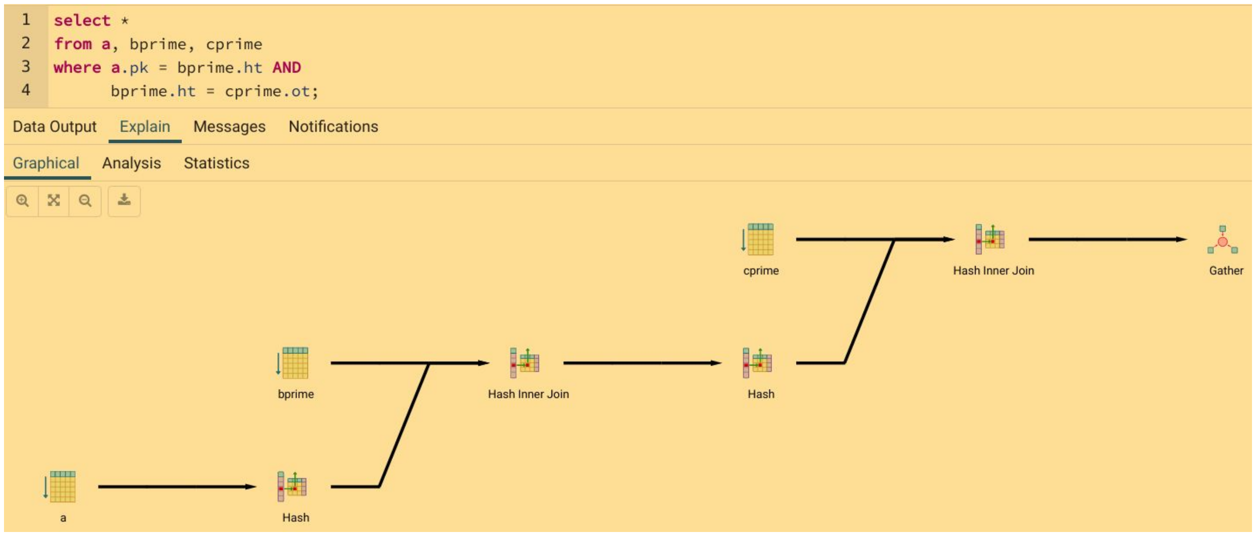


```
select *
from a, b, c
where c.ot = b.ht AND
      b.ht = a.ht;
```

Exactly the same result as 19.

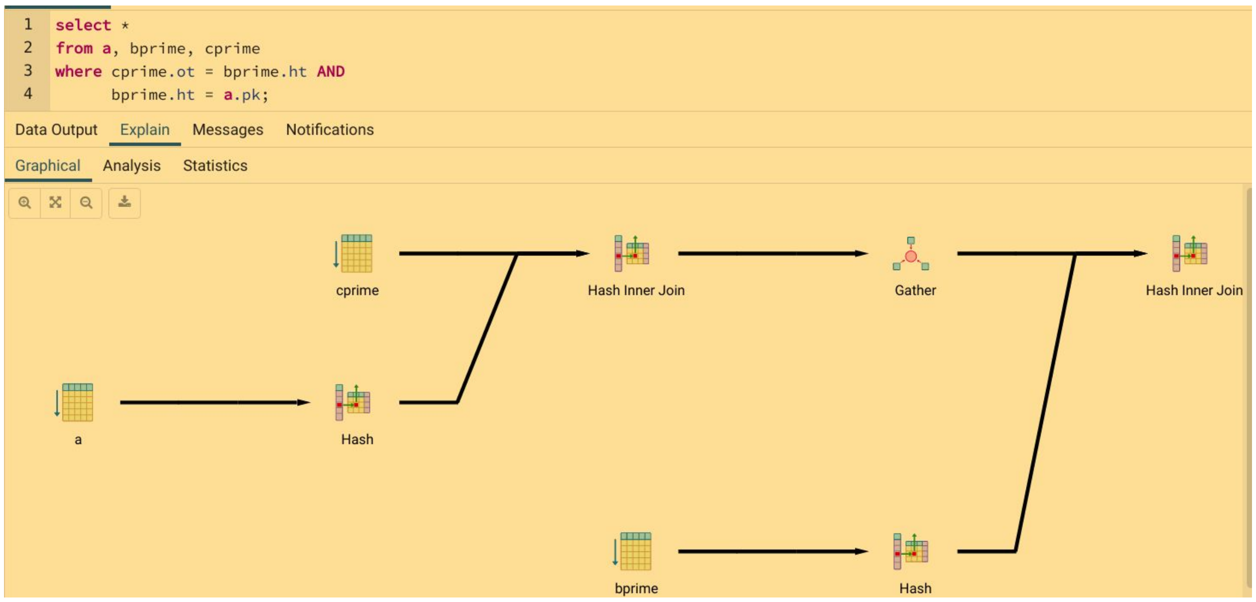
In 19 and 20, the result showed that the order of putting the relations in the query does not change the query plan chosen by the optimizer.

21.



```
select *
from a, bprime, cprime
where a.pk = bprime.ht AND
      bprime.ht = cprime.ot;
```

22.



```
select *
from a, bprime, cprime
where cprime.ot = bprime.ht AND
      bprime.ht = a.pk;
```

Hash join is used but the exact order has changed from 21.

In 21 and 22, the result showed that the actual algorithm chosen is not changed but the exact order of joining has changed. This is weird!

Part 3:

Review question 1:

Views. The SQL view construct has semantic roots in datalog. For example, suppose we have a relation ElectronicDevice(product, model, price, batteryVolume). If we want to have a “laptop” view of it, we can do

```
Create view laptop as
Select model, price, batteryVolume
From ElectronicDevice
Where product = ‘laptop’;
```

Which can also be concisely expressed by the following Datalog semantics:

```
laptop(model, price, batteryVolume) <--
        ElectronicDevice(product, model, price, batteryVolume) AND product = ‘laptop’
```

Review question 2:
Enterprise Information Integration
Business logic/workflow

Review question 3:
σ_d(Frequents ⋈ Likes ⋈ Sells)

Review question 4:
The arity of Sells = 3
The arity of Happy = 1
The arity of Likes = 2

Review question 5:
Nondistinguished variables are “bar”, “beer”, and “p”

Review question 6:
There are 6 subgoals in the CheapBeer rule;
There is 1 distinguished variable.

Review question 7:
Assume each row in table Resturant has a close time later than the open time. Otherwise, don’t input it into this relation in the first place.

(a) OpenDays(r,d) <-- Restaurant(r, d, _, _)

(b) Define a new relation Days(d) that contains 7 rows in it -- “Sunday”, “Monday”, “Tuesday”, “Wednesday”, “Thuresday”, “Friday”, “Saturday”.
ClosedDays(r,d) <- Days(d) AND NOT OpenDays(r, d)

(c) No, because (UNOs 360, Sunday) is in ClosedDays.

(d) You might, because it’s possible that UNOs 360 actually opens on Sundays but this information is not captured in the Restaurant relation. Hence the negated subgoal is not correctly computed.

Review question 8:
(a) Not safe because the distinguished variable “price” in the head does not show up anywhere in the body.

(b)

(c)