```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------
 Merge Join  (cost=0.62..1351.74 rows=8893 width=302) (actual time=0.130..45.729 rows=8897 loops=1)
   Merge Cond: (c.customerid = o.customerid)
   ->  Index Scan using customers_pkey on customers c  (cost=0.29..845.29 rows=20000 width=268) (actual time=0.032..10.950 rows=19997 loops=1)
   ->  Index Scan using ix_order_custid on orders o  (cost=0.29..345.29 rows=8893 width=30) (actual time=0.070..19.943 rows=8897 loops=1)
         Filter: (netamount > '100'::numeric)
         Rows Removed by Filter: 3103
 Planning Time: 3.843 ms
 Execution Time: 52.293 ms
```

Listing 1: merge join

```
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1033.04..1063.04 rows=12000 width=30) (actual time=9.572..10.531 rows=12000 loops=1)
   Sort Key: orderdate
   Sort Method: quicksort  Memory: 1416kB
   ->  Seq Scan on orders  (cost=0.00..220.00 rows=12000 width=30) (actual time=0.024..3.182 rows=12000 loops=1)
 Planning Time: 1.529 ms
 Execution Time: 11.672 ms
```

Listing 2: sort

```
                                                   QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=7224521.31..7224543.54 rows=8893 width=302) (actual time=111918.890..111919.623 rows=8897 loops=1)
   Sort Key: c.customerid
   Sort Method: quicksort  Memory: 2817kB
   ->  Nested Loop  (cost=0.00..7223938.00 rows=8893 width=302) (actual time=205.686..111816.823 rows=8897 loops=1)
         Join Filter: (c.customerid = o.customerid)
         Rows Removed by Join Filter: 177931103
         ->  Seq Scan on customers c  (cost=0.00..688.00 rows=20000 width=268) (actual time=8.168..31.380 rows=20000 loops=1)
         ->  Seq Scan on orders o  (cost=0.00..250.00 rows=8893 width=30) (actual time=0.006..4.604 rows=8897 loops=20000)
               Filter: (netamount > '100'::numeric)
               Rows Removed by Filter: 3103
 Planning Time: 0.777 ms
 Execution Time: 111925.339 ms
```

Listing 3: nested loop

```
                                                           QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1575.48..1590.32 rows=5934 width=304) (actual time=33.584..34.200 rows=5936 loops=1)
   Sort Key: o.netamount
   Sort Method: quicksort  Memory: 1816kB
   ->  Hash Join  (cost=938.00..1203.58 rows=5934 width=304) (actual time=12.335..23.709 rows=5936 loops=1)
         Hash Cond: (o.customerid = c.customerid)
         ->  Seq Scan on orders o  (cost=0.00..250.00 rows=5934 width=30) (actual time=0.034..5.769 rows=5936 loops=1)
               Filter: (netamount > '200'::numeric)
               Rows Removed by Filter: 6064
         ->  Hash  (cost=688.00..688.00 rows=20000 width=268) (actual time=12.254..12.254 rows=20000 loops=1)
               Buckets: 32768  Batches: 1  Memory Usage: 4108kB
               ->  Seq Scan on customers c  (cost=0.00..688.00 rows=20000 width=268) (actual time=0.012..5.910 rows=20000 loops=1)
 Planning Time: 1.154 ms
 Execution Time: 40.343 ms
```

Listing 4: hash join

```
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on customers c  (cost=0.00..788.00 rows=769 width=268) (actual time=0.101..10.838 rows=776 loops=1)
   Filter: ((age < 20) OR (age = 22))
   Rows Removed by Filter: 19224
 Planning Time: 0.233 ms
 Execution Time: 11.107 ms
```

Listing 5: multiple conditions in filter

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
Limit  (cost=0.00..0.18 rows=10 width=30) (actual time=0.033..0.041 rows=10 loops=1)
  ->  Seq Scan on orders  (cost=0.00..220.00 rows=12000 width=30) (actual time=0.030..0.034 rows=10 loops=1)
Planning Time: 0.154 ms
Execution Time: 0.414 ms
```

Listing 6: limit

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
Hash Right Join  (cost=938.00..1189.51 rows=20000 width=298) (actual time=10.990..29.612 rows=23004 loops=1)
  Hash Cond: (o.customerid = c.customerid)
  ->  Seq Scan on orders o  (cost=0.00..220.00 rows=12000 width=30) (actual time=0.010..1.664 rows=12000 loops=1)
  ->  Hash  (cost=688.00..688.00 rows=20000 width=268) (actual time=10.940..10.940 rows=20000 loops=1)
        Buckets: 32768  Batches: 1  Memory Usage: 4108kB
        ->  Seq Scan on customers c  (cost=0.00..688.00 rows=20000 width=268) (actual time=0.016..4.918 rows=20000 loops=1)
Planning Time: 0.707 ms
Execution Time: 32.227 ms
```

Listing 7: hash right join

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
Index Scan using orders_pkey on orders  (cost=0.29..2.30 rows=1 width=30) (actual time=1.361..1.366 rows=1 loops=1)
  Index Cond: (orderid = 3)
Planning Time: 0.191 ms
Execution Time: 1.673 ms
```

Listing 8: index cond

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
Hash Join  (cost=22.91..127.58 rows=974 width=500) (actual time=2.509..21.365 rows=2922 loops=1)
  Hash Cond: (a.attrelid = c.oid)
  Join Filter: has_column_privilege(c.oid, a.attnum, 'select'::text)
  ->  Seq Scan on pg_attribute a  (cost=0.00..96.61 rows=3061 width=238) (actual time=0.022..1.327 rows=2922 loops=1)
  ->  Hash  (cost=17.96..17.96 rows=396 width=266) (actual time=2.444..2.445 rows=396 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 89kB
        ->  Seq Scan on pg_class c  (cost=0.00..17.96 rows=396 width=266) (actual time=0.024..1.064 rows=396 loops=1)
Planning Time: 0.896 ms
Execution Time: 24.740 ms
```

Listing 9: non parenthesized join filter

```
                                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
Hash Join  (cost=142.19..179.59 rows=6 width=869) (actual time=14.330..15.259 rows=467 loops=1)
  Hash Cond: ((s.starelid = c.oid) AND (s.staattnum = a.attnum))
  ->  Seq Scan on pg_statistic s  (cost=0.00..32.67 rows=467 width=369) (actual time=0.023..0.133 rows=467 loops=1)
  ->  Hash  (cost=127.58..127.58 rows=974 width=504) (actual time=14.277..14.278 rows=2922 loops=1)
        Buckets: 4096 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 982kB
        ->  Hash Join  (cost=22.91..127.58 rows=974 width=504) (actual time=2.217..8.851 rows=2922 loops=1)
              Hash Cond: (a.attrelid = c.oid)
              Join Filter: has_column_privilege(c.oid, a.attnum, 'select'::text)
              ->  Seq Scan on pg_attribute a  (cost=0.00..96.61 rows=3061 width=238) (actual time=0.010..0.554 rows=2922 loops=1)
              ->  Hash  (cost=17.96..17.96 rows=396 width=266) (actual time=2.190..2.190 rows=396 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 89kB
                    ->  Seq Scan on pg_class c  (cost=0.00..17.96 rows=396 width=266) (actual time=0.023..0.944 rows=396 loops=1)
Planning Time: 2.122 ms
Execution Time: 18.919 ms
```

Listing 10: 'originally' in hash node

```
                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=143.31..182.10 rows=6 width=401) (actual time=6.445..6.501 rows=10 loops=1)
   ->  Nested Loop Left Join  (cost=143.31..182.04 rows=6 width=401) (actual time=6.443..6.496 rows=10 loops=1)
         ->  Hash Join  (cost=143.18..180.58 rows=6 width=475) (actual time=6.412..6.425 rows=10 loops=1)
               Hash Cond: ((s.starelid = c.oid) AND (s.staattnum = a.attnum))
               ->  Seq Scan on pg_statistic s  (cost=0.00..32.67 rows=467 width=349) (actual time=0.011..0.012 rows=10 loops=1)
               ->  Hash  (cost=128.57..128.57 rows=974 width=142) (actual time=6.390..6.390 rows=2922 loops=1)
                     Buckets: 4096 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 529kB
                     ->  Hash Join  (cost=23.90..128.57 rows=974 width=142) (actual time=0.394..4.966 rows=2922 loops=1)
                           Hash Cond: (a.attrelid = c.oid)
                           Join Filter: has_column_privilege(c.oid, a.attnum, 'select'::text)
                           ->  Seq Scan on pg_attribute a  (cost=0.00..96.61 rows=3061 width=70) (actual time=0.006..1.253 rows=2922 loops=1)
                                 Filter: (NOT attisdropped)
                           ->  Hash  (cost=18.95..18.95 rows=396 width=72) (actual time=0.378..0.378 rows=396 loops=1)
                                 Buckets: 1024  Batches: 1  Memory Usage: 49kB
                                 ->  Seq Scan on pg_class c  (cost=0.00..18.95 rows=396 width=72) (actual time=0.007..0.243 rows=396 loops=1)
                                       Filter: ((NOT relrowsecurity) OR (NOT row_security_active(oid)))
         ->  Index Scan using pg_namespace_oid_index on pg_namespace n  (cost=0.13..0.16 rows=1 width=68) (actual time=0.003..0.003 rows=1 loops=10)
               Index Cond: (oid = c.relnamespace)
 Planning Time: 3.761 ms
 Execution Time: 8.801 ms
```

Listing 11: left join

```
                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=250.00..250.01 rows=1 width=8) (actual time=5.717..5.717 rows=1 loops=1)
   ->  Seq Scan on orders  (cost=0.00..220.00 rows=12000 width=0) (actual time=0.025..3.280 rows=12000 loops=1)
 Planning Time: 0.172 ms
 Execution Time: 5.895 ms
```

Listing 12: aggregate

```
                                                        QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.00..1650315.00 rows=100000000 width=16) (actual time=0.241..30211.690 rows=100000000 loops=1)
   ->  Seq Scan on foo  (cost=0.00..145.00 rows=10000 width=8) (actual time=0.121..3.643 rows=10000 loops=1)
   ->  Materialize  (cost=0.00..235.00 rows=10000 width=8) (actual time=0.001..1.454 rows=10000 loops=10000)
         ->  Seq Scan on bar  (cost=0.00..145.00 rows=10000 width=8) (actual time=0.112..1.599 rows=10000 loops=1)
 Planning Time: 0.829 ms
 Execution Time: 33763.432 ms
```

Listing 13: materialize

```
                                                             QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using customers_pkey on customers  (cost=0.29..2.31 rows=1 width=4) (actual time=0.797..0.805 rows=1 loops=1)
   Index Cond: (customerid = 3)
   Heap Fetches: 1
 Planning Time: 2.832 ms
 Execution Time: 0.996 ms
```

Listing 14: index only scan

```
                                                         QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=12520.13..12542.37 rows=8893 width=302) (actual time=70.046..70.841 rows=8897 loops=1)
   Sort Key: c.customerid
   Sort Method: quicksort  Memory: 2817kB
   ->  Nested Loop  (cost=0.30..11936.82 rows=8893 width=302) (actual time=0.096..58.465 rows=8897 loops=1)
         ->  Seq Scan on orders o  (cost=0.00..250.00 rows=8893 width=30) (actual time=0.035..6.143 rows=8897 loops=1)
               Filter: (netamount > '100'::numeric)
               Rows Removed by Filter: 3103
         ->  Bitmap Heap Scan on customers c  (cost=0.30..1.31 rows=1 width=268) (actual time=0.004..0.004 rows=1 loops=8897)
               Recheck Cond: (customerid = o.customerid)
               Heap Blocks: exact=8897
               ->  Bitmap Index Scan on customers_pkey  (cost=0.00..0.30 rows=1 width=0) (actual time=0.003..0.003 rows=1 loops=8897)
                     Index Cond: (customerid = o.customerid)
 Planning Time: 0.781 ms
 Execution Time: 76.834 ms
```

Listing 15: bitmap heap scan

```
                                                         QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.57..4.62 rows=1 width=4) (actual time=0.046..0.046 rows=0 loops=1)
   ->  Index Only Scan using customers_pkey on customers c  (cost=0.29..2.31 rows=1 width=4) (actual time=0.045..0.045 rows=0 loops=1)
         Index Cond: (customerid = 1000000)
         Heap Fetches: 0
   ->  Index Only Scan using ix_order_custid on orders o  (cost=0.29..2.30 rows=1 width=4) (never executed)
         Index Cond: (customerid = 1000000)
         Heap Fetches: 0
 Planning Time: 0.152 ms
 Execution Time: 0.161 ms
```

Listing 16: never executed node

```
                                     QUERY PLAN
----------------------------------------------------------------------------------------
 Gather  (cost=1000.00..217018.43 rows=1 width=97)
   Workers Planned: 2
   ->  Parallel Seq Scan on pgbench_accounts  (cost=0.00..216018.33 rows=1 width=97)
         Filter: (filler ~~ '%x%'::text)
```

Listing 17: parallel gather

```
                                                         QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=11597.40..11597.41 rows=1 width=4) (actual time=124.383..124.383 rows=1 loops=1)
   ->  Gather  (cost=11597.19..11597.40 rows=2 width=4) (actual time=124.261..127.909 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=10597.19..10597.20 rows=1 width=4) (actual time=117.534..117.534 rows=1 loops=3)
               ->  Parallel Seq Scan on foo  (cost=0.00..9460.15 rows=454815 width=4) (actual time=0.025..67.197 rows=370000 loops=3)
 Planning Time: 0.289 ms
 Execution Time: 128.039 ms
```

Listing 18: partial aggregate

```
 Merge Join  (cost=1002.35..8595.96 rows=1 width=16) (actual time=221.121..225.824 rows=3 loops=1)
   Merge Cond: (foo.id = bar.id)
   ->  Gather Merge  (cost=1000.45..30009.29 rows=4 width=8) (actual time=221.072..221.322 rows=4 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Parallel Index Scan using foo_pkey on foo  (cost=0.43..29008.80 rows=2 width=8) (actual time=72.653..142.791 rows=3 loops=3)
               Filter: (a = ANY ('{1,2,3}'::integer[]))
               Rows Removed by Filter: 369997
   ->  Index Scan using bar_pkey on bar  (cost=0.29..318.29 rows=10000 width=8) (actual time=0.042..3.300 rows=10000 loops=1)
 Planning Time: 0.871 ms
 Execution Time: 225.952 ms
```

Listing 19: gather merge

```
                                                                QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=119887.82..232027.23 rows=55050 width=16) (actual time=2701.114..5316.387 rows=5505000 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Hash Join  (cost=118887.82..225522.23 rows=22938 width=16) (actual time=2690.280..4270.563 rows=1835000 loops=3)
         Hash Cond: (bar.id = foo.id)
         ->  Parallel Seq Scan on bar  (cost=0.00..94592.18 rows=4587518 width=8) (actual time=0.035..567.615 rows=3670000 loops=3)
         ->  Parallel Hash  (cost=118598.50..118598.50 rows=23146 width=8) (actual time=1370.159..1370.159 rows=1851667 loops=3)
               Buckets: 131072 (originally 65536)  Batches: 64 (originally 1)  Memory Usage: 4480kB
               ->  Parallel Seq Scan on foo  (cost=0.00..118598.50 rows=23146 width=8) (actual time=0.029..777.546 rows=1851667 loops=3)
                     Filter: ((a % 2) = 1)
                     Rows Removed by Filter: 1851667
 Planning Time: 0.845 ms
 Execution Time: 5554.032 ms
```

Listing 20: parallel hash join

```
                         QUERY PLAN
-------------------------------------------------------------
 Aggregate
   ->  Nested Loop
         ->  Seq Scan on tenk2
               Filter: (thousand = 0)
         ->  Gather
               Workers Planned: 4
               ->  Parallel Bitmap Heap Scan on tenk1
                     Recheck Cond: (hundred > 1)
                     ->  Bitmap Index Scan on tenk1_hundred
                           Index Cond: (hundred > 1)
```

Listing 21: parallel bitmap heap scan

```
                     QUERY PLAN
------------------------------------------------------
 Merge Join
   Merge Cond: ((((ec1_1.ff + 2) + 1)) = ec1.f1)
   ->  Merge Append
         Sort Key: (((ec1_1.ff + 2) + 1))
         ->  Index Scan using ec1_expr2 on ec1 ec1_1
         ->  Sort
               Sort Key: (((ec1_2.ff + 3) + 1))
               ->  Seq Scan on ec1 ec1_2
         ->  Index Scan using ec1_expr4 on ec1 ec1_3
   ->  Sort
         Sort Key: ec1.f1 USING <
         ->  Index Scan using ec1_pkey on ec1
               Index Cond: (ff = '42'::bigint)
(13 rows)
```

Listing 22: merge append and parenthesized expressions

```
                     QUERY PLAN
------------------------------------------------------
 Nested Loop
   ->  Index Scan using ec1_pkey on ec1
         Index Cond: (ff = '42'::bigint)
   ->  Append
         ->  Index Scan using ec1_expr2 on ec1 ec1_1
               Index Cond: (((ff + 2) + 1) = ec1.f1)
         ->  Seq Scan on ec1 ec1_2
               Filter: (((ff + 3) + 1) = ec1.f1)
         ->  Index Scan using ec1_expr4 on ec1 ec1_3
               Index Cond: ((ff + 4) = ec1.f1)
(10 rows)
```

Listing 23: append

```
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Subquery Scan on my_credit_card_usage_secure
   Filter: f_leak(my_credit_card_usage_secure.cnum)
   ->  Nested Loop
         Join Filter: (l.cid = r.cid)
         ->  Seq Scan on credit_usage r
               Filter: ((ymd >= '10-01-2011'::date) AND (ymd < '11-01-2011'::date))
         ->  Materialize
               ->  Hash Join
                     Hash Cond: (r_1.cid = l.cid)
                     ->  Seq Scan on credit_card r_1
                     ->  Hash
                           ->  Seq Scan on customer l
                                 Filter: (name = (CURRENT_USER)::text)
(13 rows)
```

Listing 24: subqyer scan

```
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Nested Loop
   ->  Hash Full Join
         Hash Cond: (COALESCE(a.q1, '0'::bigint) = COALESCE(b.q2, '-1'::bigint))
         ->  Seq Scan on int8_tbl a
         ->  Hash
               ->  Seq Scan on int8_tbl b
   ->  Index Scan using tenk1_unique2 on tenk1 c
         Index Cond: (unique2 = COALESCE((COALESCE(a.q1, '0'::bigint)), (COALESCE(b.q2, '-1'::bigint))))
(8 rows)
```

Listing 25: hash full join

```
           QUERY PLAN
--------------------------------
 Hash Left Join
   Hash Cond: (p.k = c.k)
   ->  Seq Scan on parent p
   ->  Hash
         ->  Seq Scan on child c
(5 rows)
```

Listing 26: hash left join

```
                            QUERY PLAN
------------------------------------------------------------------
 Merge Append
   Sort Key: tenk1.thousand, tenk1.tenthous
   ->  Index Only Scan using tenk1_thous_tenthous on tenk1
   ->  Sort
         Sort Key: 42, 42
         ->  Index Only Scan using tenk1_hundred on tenk1 tenk1_1
(6 rows)
```

Listing 27: multiple sort keys