GUC
German University in Cairo

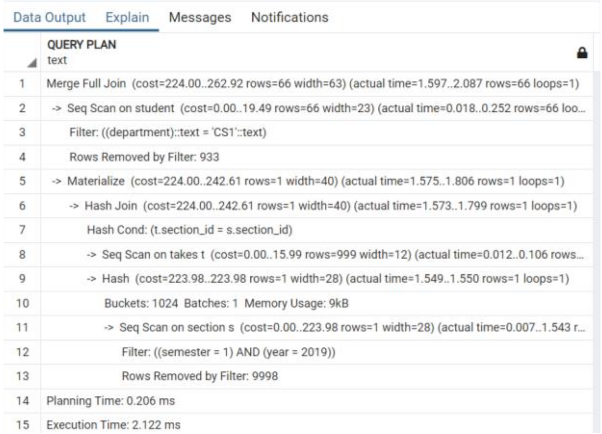
(CSEN604) Data Bases II

Supervised by: Dr. Wael Abouelsaadat

Schema 1

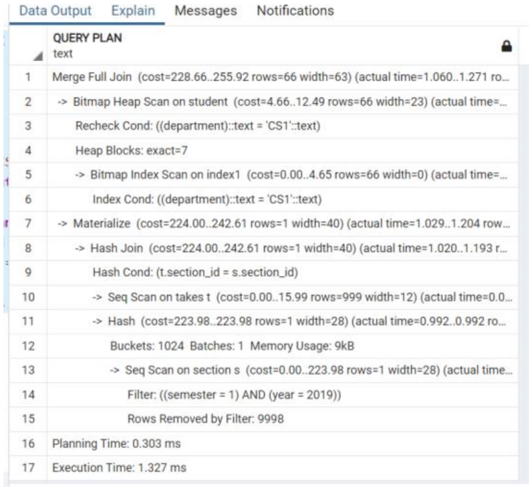
Query 1:

Planning time and execution time with no index:



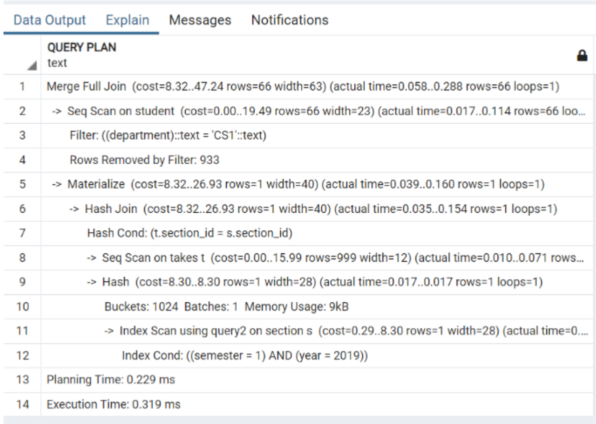
```
     QUERY PLAN
     text
1    Merge Full Join  (cost=224.00..262.92 rows=66 width=63) (actual time=1.597..2.087 rows=66 loops=1)
2      -> Seq Scan on student  (cost=0.00..19.49 rows=66 width=23) (actual time=0.018..0.252 rows=66 loo...
3          Filter: ((department)::text = 'CS1'::text)
4          Rows Removed by Filter: 933
5      -> Materialize  (cost=224.00..242.61 rows=1 width=40) (actual time=1.575..1.806 rows=1 loops=1)
6          -> Hash Join  (cost=224.00..242.61 rows=1 width=40) (actual time=1.573..1.799 rows=1 loops=1)
7              Hash Cond: (t.section_id = s.section_id)
8              -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.012..0.106 rows...
9              -> Hash  (cost=223.98..223.98 rows=1 width=28) (actual time=1.549..1.550 rows=1 loops=1)
10                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
11                 -> Seq Scan on section s  (cost=0.00..223.98 rows=1 width=28) (actual time=0.007..1.543 r...
12                     Filter: ((semester = 1) AND (year = 2019))
13                     Rows Removed by Filter: 9998
14   Planning Time: 0.206 ms
15   Execution Time: 2.122 ms
```

Planning and execution time using B+Trees on department:



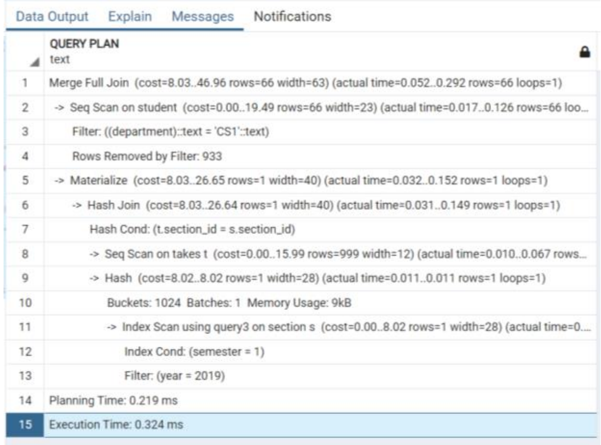
```
     QUERY PLAN
     text
1    Merge Full Join  (cost=228.66..255.92 rows=66 width=63) (actual time=1.060..1.271 ro...
2      -> Bitmap Heap Scan on student  (cost=4.66..12.49 rows=66 width=23) (actual time=...
3          Recheck Cond: ((department)::text = 'CS1'::text)
4          Heap Blocks: exact=7
5          -> Bitmap Index Scan on index1  (cost=0.00..4.65 rows=66 width=0) (actual time=...
6              Index Cond: ((department)::text = 'CS1'::text)
7      -> Materialize  (cost=224.00..242.61 rows=1 width=40) (actual time=1.029..1.204 row...
8          -> Hash Join  (cost=224.00..242.61 rows=1 width=40) (actual time=1.020..1.193 r...
9              Hash Cond: (t.section_id = s.section_id)
10             -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.0...
11             -> Hash  (cost=223.98..223.98 rows=1 width=28) (actual time=0.992..0.992 ro...
12                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
13                 -> Seq Scan on section s  (cost=0.00..223.98 rows=1 width=28) (actual time...
14                     Filter: ((semester = 1) AND (year = 2019))
15                     Rows Removed by Filter: 9998
16   Planning Time: 0.303 ms
17   Execution Time: 1.327 ms
```

Planning and execution time using B+Trees on year and semester:



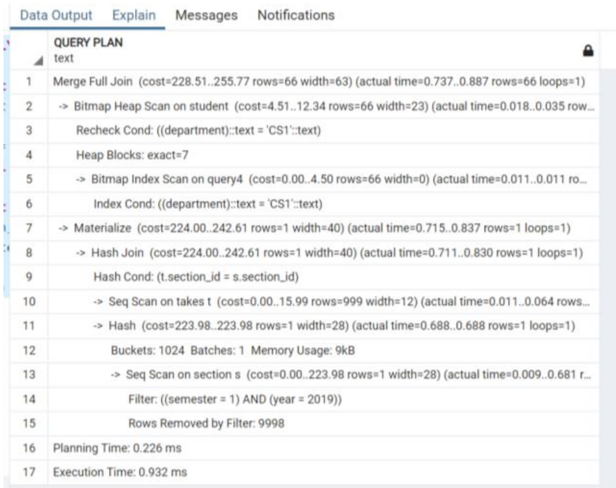
```
   QUERY PLAN
   text
1  Merge Full Join  (cost=8.32..47.24 rows=66 width=63) (actual time=0.058..0.288 rows=66 loops=1)
2    -> Seq Scan on student  (cost=0.00..19.49 rows=66 width=23) (actual time=0.017..0.114 rows=66 loo...
3        Filter: ((department)::text = 'CS1'::text)
4        Rows Removed by Filter: 933
5    -> Materialize  (cost=8.32..26.93 rows=1 width=40) (actual time=0.039..0.160 rows=1 loops=1)
6        -> Hash Join  (cost=8.32..26.93 rows=1 width=40) (actual time=0.035..0.154 rows=1 loops=1)
7            Hash Cond: (t.section_id = s.section_id)
8            -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.010..0.071 rows...
9            -> Hash  (cost=8.30..8.30 rows=1 width=28) (actual time=0.017..0.017 rows=1 loops=1)
10               Buckets: 1024  Batches: 1  Memory Usage: 9kB
11               -> Index Scan using query2 on section s  (cost=0.29..8.30 rows=1 width=28) (actual time=0....
12                   Index Cond: ((semester = 1) AND (year = 2019))
13 Planning Time: 0.229 ms
14 Execution Time: 0.319 ms
```

Planning and execution time using hash on semester:



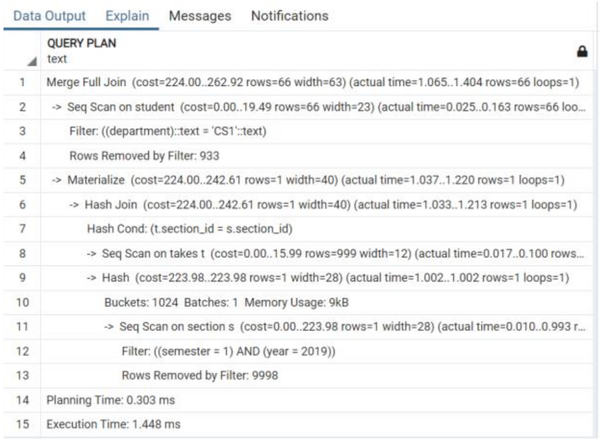
```
   QUERY PLAN
   text
1  Merge Full Join  (cost=8.03..46.96 rows=66 width=63) (actual time=0.052..0.292 rows=66 loops=1)
2    -> Seq Scan on student  (cost=0.00..19.49 rows=66 width=23) (actual time=0.017..0.126 rows=66 loo...
3        Filter: ((department)::text = 'CS1'::text)
4        Rows Removed by Filter: 933
5    -> Materialize  (cost=8.03..26.65 rows=1 width=40) (actual time=0.032..0.152 rows=1 loops=1)
6        -> Hash Join  (cost=8.03..26.64 rows=1 width=40) (actual time=0.031..0.149 rows=1 loops=1)
7            Hash Cond: (t.section_id = s.section_id)
8            -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.010..0.067 rows...
9            -> Hash  (cost=8.02..8.02 rows=1 width=28) (actual time=0.011..0.011 rows=1 loops=1)
10               Buckets: 1024  Batches: 1  Memory Usage: 9kB
11               -> Index Scan using query3 on section s  (cost=0.00..8.02 rows=1 width=28) (actual time=0....
12                   Index Cond: (semester = 1)
13                   Filter: (year = 2019)
14 Planning Time: 0.219 ms
15 Execution Time: 0.324 ms
```

Using hash on department:



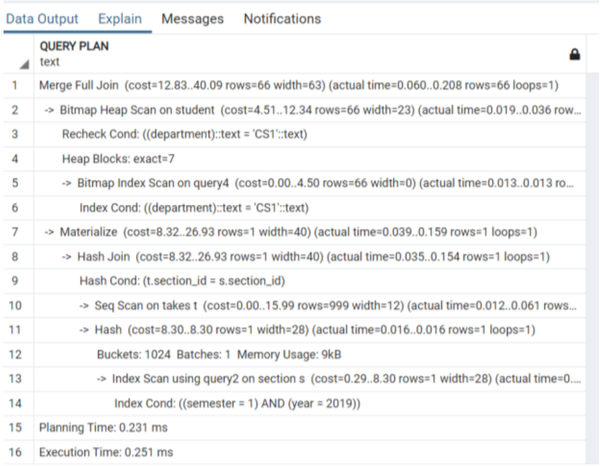
```
   QUERY PLAN
   text
1  Merge Full Join  (cost=228.51..255.77 rows=66 width=63) (actual time=0.737..0.887 rows=66 loops=1)
2    -> Bitmap Heap Scan on student  (cost=4.51..12.34 rows=66 width=23) (actual time=0.018..0.035 row...
3        Recheck Cond: ((department)::text = 'CS1'::text)
4        Heap Blocks: exact=7
5        -> Bitmap Index Scan on query4  (cost=0.00..4.50 rows=66 width=0) (actual time=0.011..0.011 ro...
6            Index Cond: ((department)::text = 'CS1'::text)
7    -> Materialize  (cost=224.00..242.61 rows=1 width=40) (actual time=0.715..0.837 rows=1 loops=1)
8        -> Hash Join  (cost=224.00..242.61 rows=1 width=40) (actual time=0.711..0.830 rows=1 loops=1)
9            Hash Cond: (t.section_id = s.section_id)
10           -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.011..0.064 rows...
11           -> Hash  (cost=223.98..223.98 rows=1 width=28) (actual time=0.688..0.688 rows=1 loops=1)
12               Buckets: 1024  Batches: 1  Memory Usage: 9kB
13               -> Seq Scan on section s  (cost=0.00..223.98 rows=1 width=28) (actual time=0.009..0.681 r...
14                   Filter: ((semester = 1) AND (year = 2019))
15                   Rows Removed by Filter: 9998
16 Planning Time: 0.226 ms
17 Execution Time: 0.932 ms
```

Using hash on year:



```
     QUERY PLAN
     text
1    Merge Full Join  (cost=224.00..262.92 rows=66 width=63) (actual time=1.065..1.404 rows=66 loops=1)
2      -> Seq Scan on student  (cost=0.00..19.49 rows=66 width=23) (actual time=0.025..0.163 rows=66 loo...
3          Filter: ((department)::text = 'CS1'::text)
4          Rows Removed by Filter: 933
5      -> Materialize  (cost=224.00..242.61 rows=1 width=40) (actual time=1.037..1.220 rows=1 loops=1)
6          -> Hash Join  (cost=224.00..242.61 rows=1 width=40) (actual time=1.033..1.213 rows=1 loops=1)
7              Hash Cond: (t.section_id = s.section_id)
8              -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.017..0.100 rows...
9              -> Hash  (cost=223.98..223.98 rows=1 width=28) (actual time=1.002..1.002 rows=1 loops=1)
10                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
11                 -> Seq Scan on section s  (cost=0.00..223.98 rows=1 width=28) (actual time=0.010..0.993 r...
12                     Filter: ((semester = 1) AND (year = 2019))
13                     Rows Removed by Filter: 9998
14   Planning Time: 0.303 ms
15   Execution Time: 1.448 ms
```

Planning and execution time using mixed indices B+tree on year and semester and hash on department:



```
     QUERY PLAN
     text
1    Merge Full Join  (cost=12.83..40.09 rows=66 width=63) (actual time=0.060..0.208 rows=66 loops=1)
2      -> Bitmap Heap Scan on student  (cost=4.51..12.34 rows=66 width=23) (actual time=0.019..0.036 row...
3          Recheck Cond: ((department)::text = 'CS1'::text)
4          Heap Blocks: exact=7
5          -> Bitmap Index Scan on query4  (cost=0.00..4.50 rows=66 width=0) (actual time=0.013..0.013 ro...
6              Index Cond: ((department)::text = 'CS1'::text)
7      -> Materialize  (cost=8.32..26.93 rows=1 width=40) (actual time=0.039..0.159 rows=1 loops=1)
8          -> Hash Join  (cost=8.32..26.93 rows=1 width=40) (actual time=0.035..0.154 rows=1 loops=1)
9              Hash Cond: (t.section_id = s.section_id)
10             -> Seq Scan on takes t  (cost=0.00..15.99 rows=999 width=12) (actual time=0.012..0.061 rows...
11             -> Hash  (cost=8.30..8.30 rows=1 width=28) (actual time=0.016..0.016 rows=1 loops=1)
12                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
13                 -> Index Scan using query2 on section s  (cost=0.29..8.30 rows=1 width=28) (actual time=0....
14                     Index Cond: ((semester = 1) AND (year = 2019))
15   Planning Time: 0.231 ms
16   Execution Time: 0.251 ms
```

Report Summary:

By checking the results shown above we can deduce that the execution time significantly improved when using indices. Best option would be mix between B+tree on year and semester and hash on department so that all columns we are performing a query on are indexed. B+ tree runs faster when on year and semester or even on of them only as B+tree is better used on numerical data. Hash is insignificantly slower than B+tree.

Without index: 2.122 ms

With B+tree on department: 1.327 ms

With B+tree on year and semester: 0.319 ms

With hash on semester: 0.324ms

With hash on department: 0.932ms

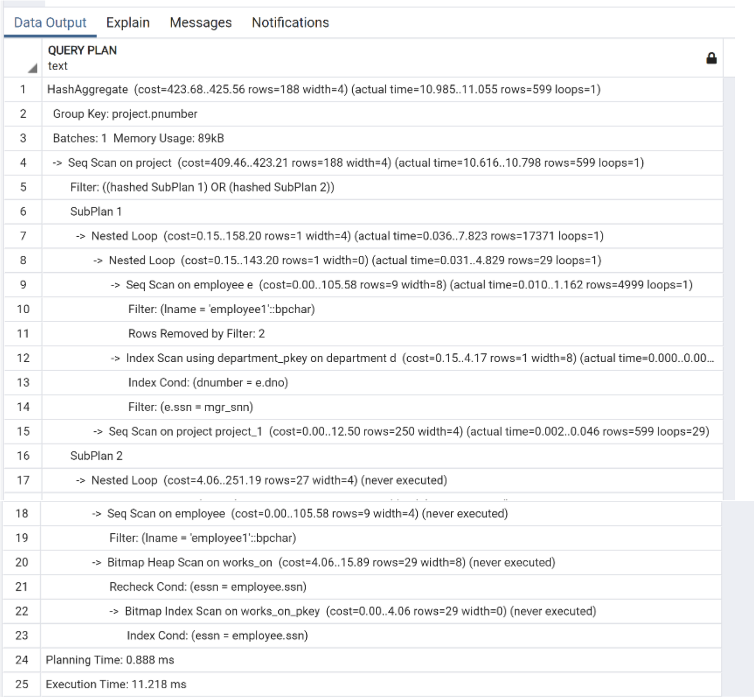
With hash on year: 1.448ms

With mixed indices B+tree on year and semester and hash on department:0.251ms

Schema 2

Query 2

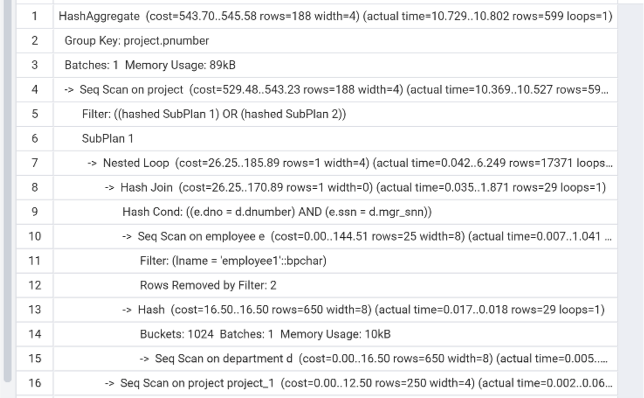
Scenario 1(No index)



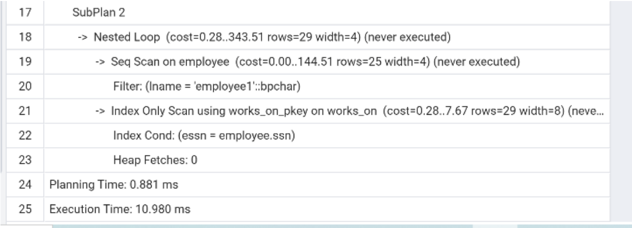
```
     QUERY PLAN
     text
1    HashAggregate  (cost=423.68..425.56 rows=188 width=4) (actual time=10.985..11.055 rows=599 loops=1)
2     Group Key: project.pnumber
3     Batches: 1  Memory Usage: 89kB
4     ->  Seq Scan on project  (cost=409.46..423.21 rows=188 width=4) (actual time=10.616..10.798 rows=599 loops=1)
5           Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
6           SubPlan 1
7             ->  Nested Loop  (cost=0.15..158.20 rows=1 width=4) (actual time=0.036..7.823 rows=17371 loops=1)
8                   ->  Nested Loop  (cost=0.15..143.20 rows=1 width=0) (actual time=0.031..4.829 rows=29 loops=1)
9                         ->  Seq Scan on employee e  (cost=0.00..105.58 rows=9 width=8) (actual time=0.010..1.162 rows=4999 loops=1)
10                              Filter: (lname = 'employee1'::bpchar)
11                              Rows Removed by Filter: 2
12                        ->  Index Scan using department_pkey on department d  (cost=0.15..4.17 rows=1 width=8) (actual time=0.000..0.00...
13                              Index Cond: (dnumber = e.dno)
14                              Filter: (e.ssn = mgr_snn)
15                  ->  Seq Scan on project project_1  (cost=0.00..12.50 rows=250 width=4) (actual time=0.002..0.046 rows=599 loops=29)
16          SubPlan 2
17            ->  Nested Loop  (cost=4.06..251.19 rows=27 width=4) (never executed)
18                  ->  Seq Scan on employee  (cost=0.00..105.58 rows=9 width=4) (never executed)
19                        Filter: (lname = 'employee1'::bpchar)
20                  ->  Bitmap Heap Scan on works_on  (cost=4.06..15.89 rows=29 width=8) (never executed)
21                        Recheck Cond: (essn = employee.ssn)
22                        ->  Bitmap Index Scan on works_on_pkey  (cost=0.00..4.06 rows=29 width=0) (never executed)
23                              Index Cond: (essn = employee.ssn)
24   Planning Time: 0.888 ms
25   Execution Time: 11.218 ms
```

Scenario 2(B+ Tree index on Ssn in Employee)

"Used B+ index on ssn because it is accessed in every loop that happens in the query, thus shortening the time"



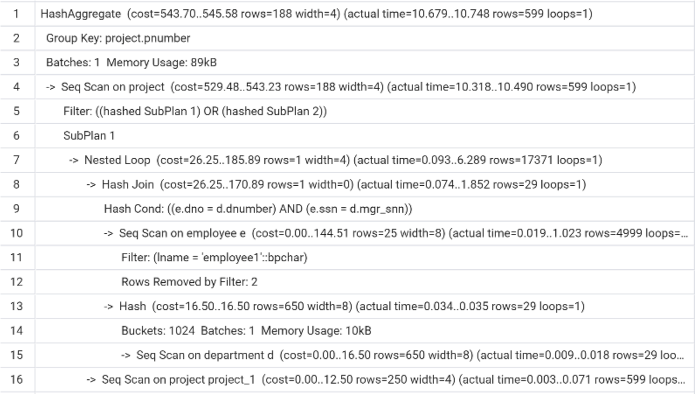
```
1    HashAggregate  (cost=543.70..545.58 rows=188 width=4) (actual time=10.729..10.802 rows=599 loops=1)
2     Group Key: project.pnumber
3     Batches: 1  Memory Usage: 89kB
4     ->  Seq Scan on project  (cost=529.48..543.23 rows=188 width=4) (actual time=10.369..10.527 rows=59...
5           Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
6           SubPlan 1
7             ->  Nested Loop  (cost=26.25..185.89 rows=1 width=4) (actual time=0.042..6.249 rows=17371 loops...
8                   ->  Hash Join  (cost=26.25..170.89 rows=1 width=0) (actual time=0.035..1.871 rows=29 loops=1)
9                         Hash Cond: ((e.dno = d.dnumber) AND (e.ssn = d.mgr_snn))
10                        ->  Seq Scan on employee e  (cost=0.00..144.51 rows=25 width=8) (actual time=0.007..1.041 ...
11                              Filter: (lname = 'employee1'::bpchar)
12                              Rows Removed by Filter: 2
13                        ->  Hash  (cost=16.50..16.50 rows=650 width=8) (actual time=0.017..0.018 rows=29 loops=1)
14                              Buckets: 1024  Batches: 1  Memory Usage: 10kB
15                              ->  Seq Scan on department d  (cost=0.00..16.50 rows=650 width=8) (actual time=0.005.....
16                  ->  Seq Scan on project project_1  (cost=0.00..12.50 rows=250 width=4) (actual time=0.002..0.06...
```



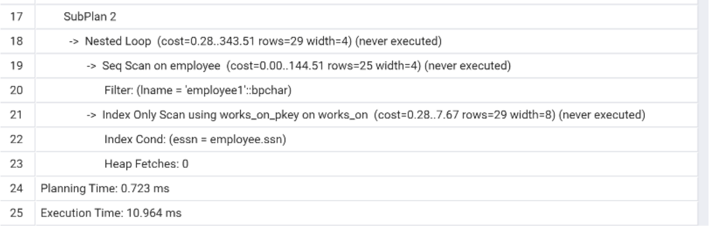
```
17        SubPlan 2
18          ->  Nested Loop  (cost=0.28..343.51 rows=29 width=4) (never executed)
19                ->  Seq Scan on employee  (cost=0.00..144.51 rows=25 width=4) (never executed)
20                      Filter: (lname = 'employee1'::bpchar)
21                ->  Index Only Scan using works_on_pkey on works_on  (cost=0.28..7.67 rows=29 width=8) (neve...
22                      Index Cond: (essn = employee.ssn)
23                      Heap Fetches: 0
24  Planning Time: 0.881 ms
25  Execution Time: 10.980 ms
```

Scenario 3(Hash index on Ssn in Employee)

“Used Hash index on ssn because it is accessed in every loop that happens in the query, thus shortening the time”



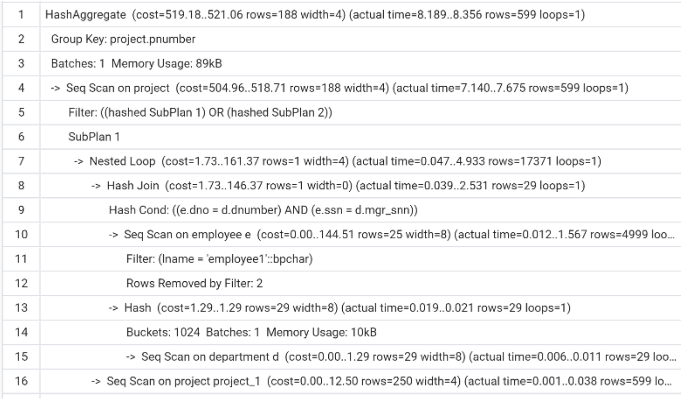
```
1   HashAggregate  (cost=543.70..545.58 rows=188 width=4) (actual time=10.679..10.748 rows=599 loops=1)
2     Group Key: project.pnumber
3     Batches: 1  Memory Usage: 89kB
4     ->  Seq Scan on project  (cost=529.48..543.23 rows=188 width=4) (actual time=10.318..10.490 rows=599 loops=1)
5           Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
6           SubPlan 1
7             ->  Nested Loop  (cost=26.25..185.89 rows=1 width=4) (actual time=0.093..6.289 rows=17371 loops=1)
8                   ->  Hash Join  (cost=26.25..170.89 rows=1 width=0) (actual time=0.074..1.852 rows=29 loops=1)
9                         Hash Cond: ((e.dno = d.dnumber) AND (e.ssn = d.mgr_snn))
10                        ->  Seq Scan on employee e  (cost=0.00..144.51 rows=25 width=8) (actual time=0.019..1.023 rows=4999 loops=...
11                              Filter: (lname = 'employee1'::bpchar)
12                              Rows Removed by Filter: 2
13                        ->  Hash  (cost=16.50..16.50 rows=650 width=8) (actual time=0.034..0.035 rows=29 loops=1)
14                              Buckets: 1024  Batches: 1  Memory Usage: 10kB
15                              ->  Seq Scan on department d  (cost=0.00..16.50 rows=650 width=8) (actual time=0.009..0.018 rows=29 loo...
16                  ->  Seq Scan on project project_1  (cost=0.00..12.50 rows=250 width=4) (actual time=0.003..0.071 rows=599 loops...
```



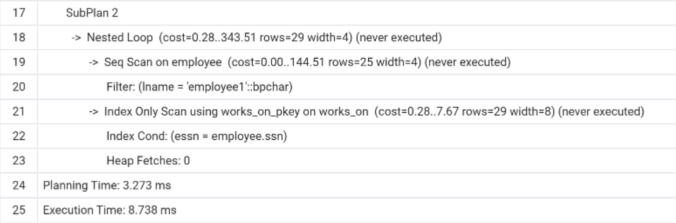
```
17        SubPlan 2
18          ->  Nested Loop  (cost=0.28..343.51 rows=29 width=4) (never executed)
19                ->  Seq Scan on employee  (cost=0.00..144.51 rows=25 width=4) (never executed)
20                      Filter: (lname = 'employee1'::bpchar)
21                ->  Index Only Scan using works_on_pkey on works_on  (cost=0.28..7.67 rows=29 width=8) (never executed)
22                      Index Cond: (essn = employee.ssn)
23                      Heap Fetches: 0
24  Planning Time: 0.723 ms
25  Execution Time: 10.964 ms
```

Scenario 4(Mixed index [Hash on Ssn, B+ on Dnumber on Department])

“I used here the a hash index on ssn because it is accessed at everyloop and also the dnumber is accessed many times so I used B+ index with it thus the time decreased drastically compared to the 3 previous runs”



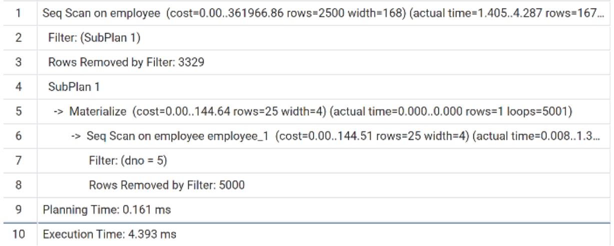
```
1   HashAggregate  (cost=519.18..521.06 rows=188 width=4) (actual time=8.189..8.356 rows=599 loops=1)
2     Group Key: project.pnumber
3     Batches: 1  Memory Usage: 89kB
4     ->  Seq Scan on project  (cost=504.96..518.71 rows=188 width=4) (actual time=7.140..7.675 rows=599 loops=1)
5           Filter: ((hashed SubPlan 1) OR (hashed SubPlan 2))
6           SubPlan 1
7             ->  Nested Loop  (cost=1.73..161.37 rows=1 width=4) (actual time=0.047..4.933 rows=17371 loops=1)
8                   ->  Hash Join  (cost=1.73..146.37 rows=1 width=0) (actual time=0.039..2.531 rows=29 loops=1)
9                         Hash Cond: ((e.dno = d.dnumber) AND (e.ssn = d.mgr_snn))
10                        ->  Seq Scan on employee e  (cost=0.00..144.51 rows=25 width=8) (actual time=0.012..1.567 rows=4999 loo...
11                              Filter: (lname = 'employee1'::bpchar)
12                              Rows Removed by Filter: 2
13                        ->  Hash  (cost=1.29..1.29 rows=29 width=8) (actual time=0.019..0.021 rows=29 loops=1)
14                              Buckets: 1024  Batches: 1  Memory Usage: 10kB
15                              ->  Seq Scan on department d  (cost=0.00..1.29 rows=29 width=8) (actual time=0.006..0.011 rows=29 loo...
16                  ->  Seq Scan on project project_1  (cost=0.00..12.50 rows=250 width=4) (actual time=0.001..0.038 rows=599 lo...
```



```
    SubPlan 2
      ->  Nested Loop  (cost=0.28..343.51 rows=29 width=4) (never executed)
            ->  Seq Scan on employee  (cost=0.00..144.51 rows=25 width=4) (never executed)
                  Filter: (lname = 'employee1'::bpchar)
            ->  Index Only Scan using works_on_pkey on works_on  (cost=0.28..7.67 rows=29 width=8) (never executed)
                  Index Cond: (essn = employee.ssn)
                  Heap Fetches: 0
Planning Time: 3.273 ms
Execution Time: 8.738 ms
```

<u>Query 3</u>

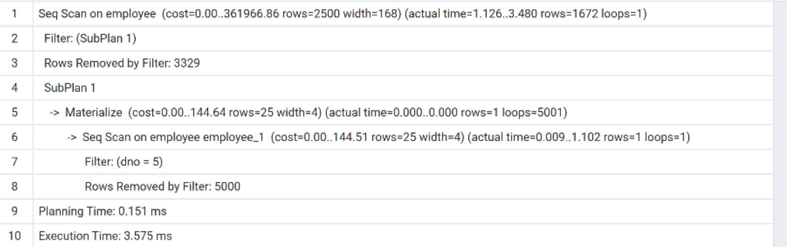
Scenario 1(No index)



```
Seq Scan on employee  (cost=0.00..361966.86 rows=2500 width=168) (actual time=1.405..4.287 rows=167...
  Filter: (SubPlan 1)
  Rows Removed by Filter: 3329
  SubPlan 1
    ->  Materialize  (cost=0.00..144.64 rows=25 width=4) (actual time=0.000..0.000 rows=1 loops=5001)
          ->  Seq Scan on employee employee_1  (cost=0.00..144.51 rows=25 width=4) (actual time=0.008..1.3...
                Filter: (dno = 5)
                Rows Removed by Filter: 5000
Planning Time: 0.161 ms
Execution Time: 4.393 ms
```

Scenario 2(B+ Tree index on Dnumber in Department)

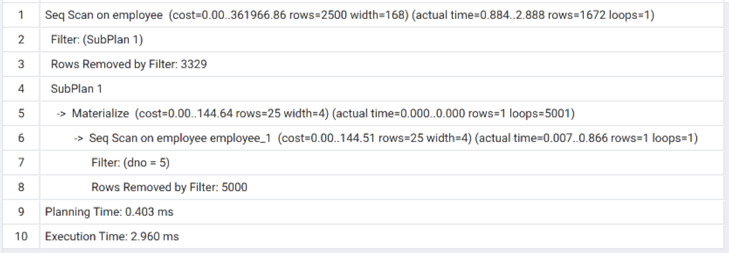
“All the retrieved events should have a salary of those who have the dnumber=5, thus ordering by using the B+ index, the dnumber will surely make the execution faster”



```
Seq Scan on employee  (cost=0.00..361966.86 rows=2500 width=168) (actual time=1.126..3.480 rows=1672 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 3329
  SubPlan 1
    ->  Materialize  (cost=0.00..144.64 rows=25 width=4) (actual time=0.000..0.000 rows=1 loops=5001)
          ->  Seq Scan on employee employee_1  (cost=0.00..144.51 rows=25 width=4) (actual time=0.009..1.102 rows=1 loops=1)
                Filter: (dno = 5)
                Rows Removed by Filter: 5000
Planning Time: 0.151 ms
Execution Time: 3.575 ms
```

Scenario 3(Hash index on Salary in Employee)

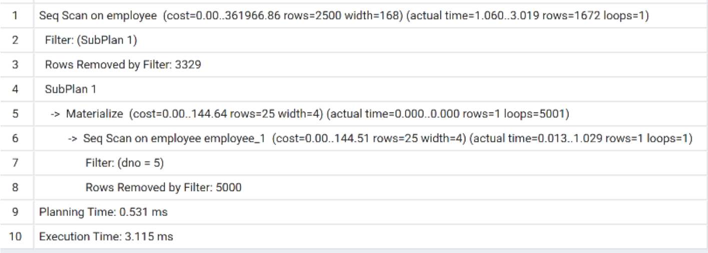
“All the retrieved employees should have salary>than all the salaries of those who have d=no5, thus ordering by using the Hash index, the salary will surely make the execution faster”



```
Seq Scan on employee  (cost=0.00..361966.86 rows=2500 width=168) (actual time=0.884..2.888 rows=1672 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 3329
  SubPlan 1
    ->  Materialize  (cost=0.00..144.64 rows=25 width=4) (actual time=0.000..0.000 rows=1 loops=5001)
          ->  Seq Scan on employee employee_1  (cost=0.00..144.51 rows=25 width=4) (actual time=0.007..0.866 rows=1 loops=1)
                Filter: (dno = 5)
                Rows Removed by Filter: 5000
Planning Time: 0.403 ms
Execution Time: 2.960 ms
```

Scenario 4(Mixed index[B+ on dnumber,Hash on salary])

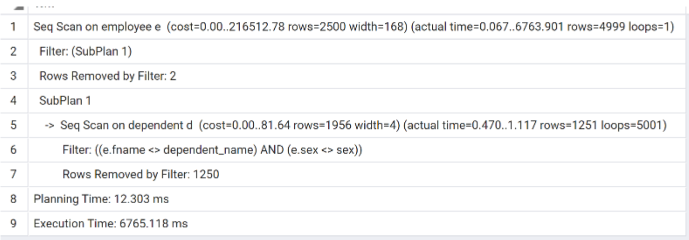
“Using both indexes on dnumber the B+ and on the salary using the Hash resulted in a shorter execution time than the original query”



```
Seq Scan on employee  (cost=0.00..361966.86 rows=2500 width=168) (actual time=1.060..3.019 rows=1672 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 3329
  SubPlan 1
    ->  Materialize  (cost=0.00..144.64 rows=25 width=4) (actual time=0.000..0.000 rows=1 loops=5001)
          ->  Seq Scan on employee employee_1  (cost=0.00..144.51 rows=25 width=4) (actual time=0.013..1.029 rows=1 loops=1)
                Filter: (dno = 5)
                Rows Removed by Filter: 5000
Planning Time: 0.531 ms
Execution Time: 3.115 ms
```

Query 4

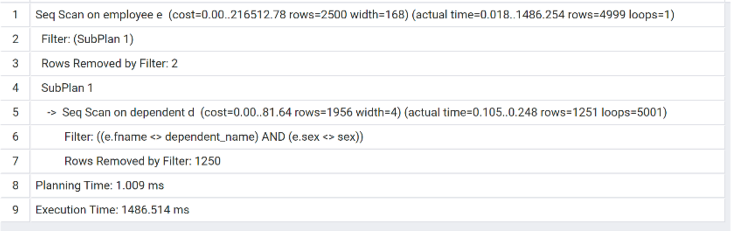
Scenario 1(No index)



```
Seq Scan on employee e  (cost=0.00..216512.78 rows=2500 width=168) (actual time=0.067..6763.901 rows=4999 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 2
  SubPlan 1
    ->  Seq Scan on dependent d  (cost=0.00..81.64 rows=1956 width=4) (actual time=0.470..1.117 rows=1251 loops=5001)
          Filter: ((e.fname <> dependent_name) AND (e.sex <> sex))
          Rows Removed by Filter: 1250
Planning Time: 12.303 ms
Execution Time: 6765.118 ms
```

Scenario 2(B+ Tree index on fname in Employee)

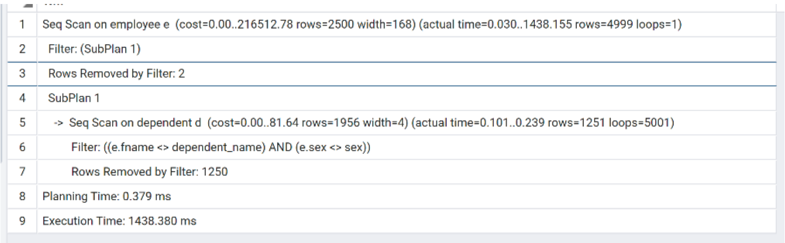
“fname is always compared with the dependent name so using the B+ index on the fname will make the execution time less”



```
Seq Scan on employee e  (cost=0.00..216512.78 rows=2500 width=168) (actual time=0.018..1486.254 rows=4999 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 2
  SubPlan 1
    ->  Seq Scan on dependent d  (cost=0.00..81.64 rows=1956 width=4) (actual time=0.105..0.248 rows=1251 loops=5001)
          Filter: ((e.fname <> dependent_name) AND (e.sex <> sex))
          Rows Removed by Filter: 1250
Planning Time: 1.009 ms
Execution Time: 1486.514 ms
```

Scenario 3(Hash index on sex in Employee)

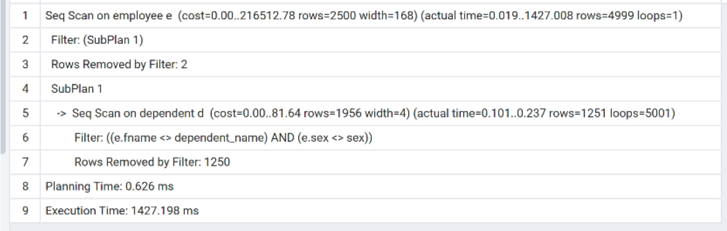
“the sex of the employee is always compared so using also a hash index with will make the execution tim less”



```
Seq Scan on employee e  (cost=0.00..216512.78 rows=2500 width=168) (actual time=0.030..1438.155 rows=4999 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 2
  SubPlan 1
    ->  Seq Scan on dependent d  (cost=0.00..81.64 rows=1956 width=4) (actual time=0.101..0.239 rows=1251 loops=5001)
          Filter: ((e.fname <> dependent_name) AND (e.sex <> sex))
          Rows Removed by Filter: 1250
Planning Time: 0.379 ms
Execution Time: 1438.380 ms
```

Scenario 4(Mixed index[B+ on fname, Hash on sex])

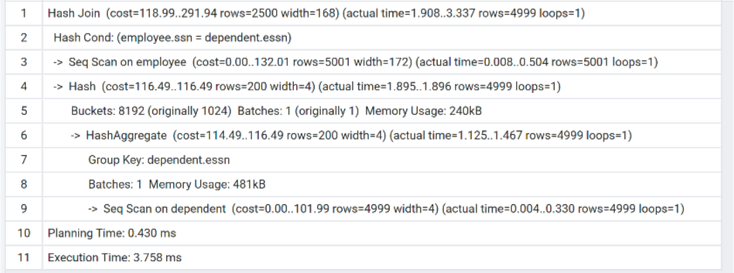
“Using 2 index increased the planning time but decreased the overall execution time and resulted in the best performance”



```
Seq Scan on employee e  (cost=0.00..216512.78 rows=2500 width=168) (actual time=0.019..1427.008 rows=4999 loops=1)
  Filter: (SubPlan 1)
  Rows Removed by Filter: 2
  SubPlan 1
    ->  Seq Scan on dependent d  (cost=0.00..81.64 rows=1956 width=4) (actual time=0.101..0.237 rows=1251 loops=5001)
          Filter: ((e.fname <> dependent_name) AND (e.sex <> sex))
          Rows Removed by Filter: 1250
Planning Time: 0.626 ms
Execution Time: 1427.198 ms
```

Query 5

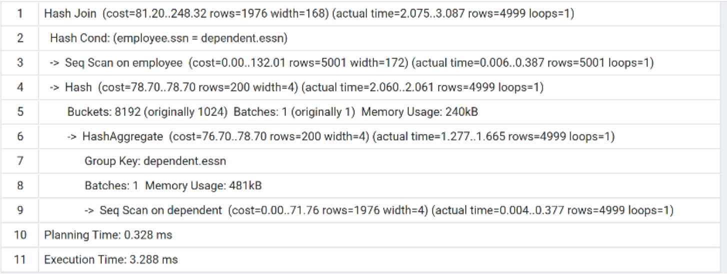
Scenario 1(No index)



```
Hash Join  (cost=118.99..291.94 rows=2500 width=168) (actual time=1.908..3.337 rows=4999 loops=1)
  Hash Cond: (employee.ssn = dependent.essn)
  ->  Seq Scan on employee  (cost=0.00..132.01 rows=5001 width=172) (actual time=0.008..0.504 rows=5001 loops=1)
  ->  Hash  (cost=116.49..116.49 rows=200 width=4) (actual time=1.895..1.896 rows=4999 loops=1)
        Buckets: 8192 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 240kB
        ->  HashAggregate  (cost=114.49..116.49 rows=200 width=4) (actual time=1.125..1.467 rows=4999 loops=1)
              Group Key: dependent.essn
              Batches: 1  Memory Usage: 481kB
              ->  Seq Scan on dependent  (cost=0.00..101.99 rows=4999 width=4) (actual time=0.004..0.330 rows=4999 loops=1)
Planning Time: 0.430 ms
Execution Time: 3.758 ms
```

Scenario 2(B+ Tree index on Ssn in Employee)

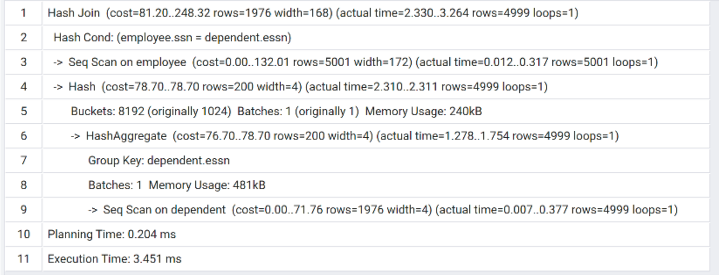
“The query only check on the ssn in employee and compare it with essn in dependent so creating a B+ index on ssn normally decreased the time”



```
Hash Join  (cost=81.20..248.32 rows=1976 width=168) (actual time=2.075..3.087 rows=4999 loops=1)
  Hash Cond: (employee.ssn = dependent.essn)
  ->  Seq Scan on employee  (cost=0.00..132.01 rows=5001 width=172) (actual time=0.006..0.387 rows=5001 loops=1)
  ->  Hash  (cost=78.70..78.70 rows=200 width=4) (actual time=2.060..2.061 rows=4999 loops=1)
        Buckets: 8192 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 240kB
        ->  HashAggregate  (cost=76.70..78.70 rows=200 width=4) (actual time=1.277..1.665 rows=4999 loops=1)
              Group Key: dependent.essn
              Batches: 1  Memory Usage: 481kB
              ->  Seq Scan on dependent  (cost=0.00..71.76 rows=1976 width=4) (actual time=0.004..0.377 rows=4999 loops=1)
Planning Time: 0.328 ms
Execution Time: 3.288 ms
```

Scenario 3(Hash index on essn in dependent)

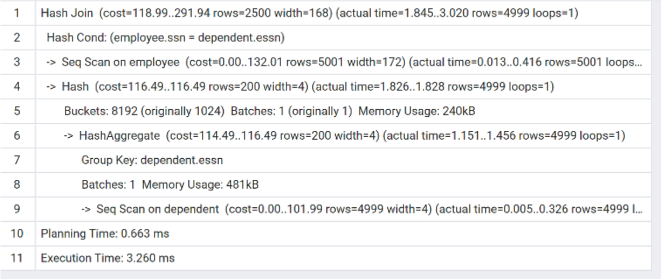
“The query only check on the ssn in employee and compare it with essn in dependent so creating a B+ index on essn normally decreased the time”



```
Hash Join  (cost=81.20..248.32 rows=1976 width=168) (actual time=2.330..3.264 rows=4999 loops=1)
  Hash Cond: (employee.ssn = dependent.essn)
  ->  Seq Scan on employee  (cost=0.00..132.01 rows=5001 width=172) (actual time=0.012..0.317 rows=5001 loops=1)
  ->  Hash  (cost=78.70..78.70 rows=200 width=4) (actual time=2.310..2.311 rows=4999 loops=1)
        Buckets: 8192 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 240kB
        ->  HashAggregate  (cost=76.70..78.70 rows=200 width=4) (actual time=1.278..1.754 rows=4999 loops=1)
              Group Key: dependent.essn
              Batches: 1  Memory Usage: 481kB
              ->  Seq Scan on dependent  (cost=0.00..71.76 rows=1976 width=4) (actual time=0.007..0.377 rows=4999 loops=1)
Planning Time: 0.204 ms
Execution Time: 3.451 ms
```

Scenario 4(Mixed index[B+ on SSn,Hash on essn])

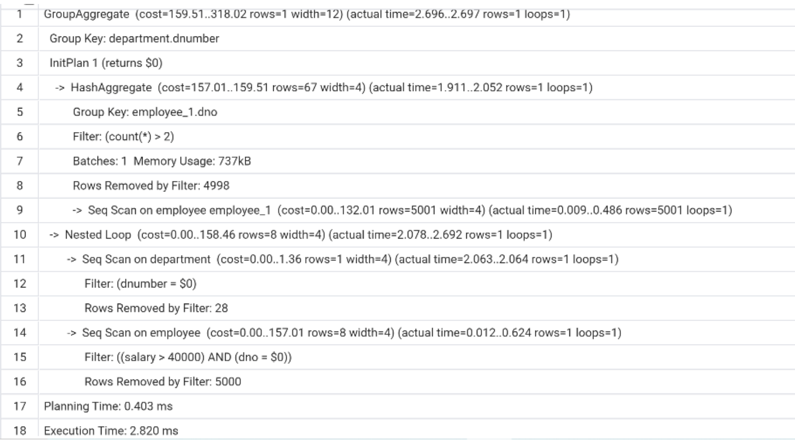
“The query only check on the ssn in employee and compare it with essn in dependent so creating a B+ index on ssn and essn resulted in the least execution time”



```
Hash Join  (cost=118.99..291.94 rows=2500 width=168) (actual time=1.845..3.020 rows=4999 loops=1)
  Hash Cond: (employee.ssn = dependent.essn)
  ->  Seq Scan on employee  (cost=0.00..132.01 rows=5001 width=172) (actual time=0.013..0.416 rows=5001 loops...
  ->  Hash  (cost=116.49..116.49 rows=200 width=4) (actual time=1.826..1.828 rows=4999 loops=1)
        Buckets: 8192 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 240kB
        ->  HashAggregate  (cost=114.49..116.49 rows=200 width=4) (actual time=1.151..1.456 rows=4999 loops=1)
              Group Key: dependent.essn
              Batches: 1  Memory Usage: 481kB
              ->  Seq Scan on dependent  (cost=0.00..101.99 rows=4999 width=4) (actual time=0.005..0.326 rows=4999 l...
Planning Time: 0.663 ms
Execution Time: 3.260 ms
```

Query 6

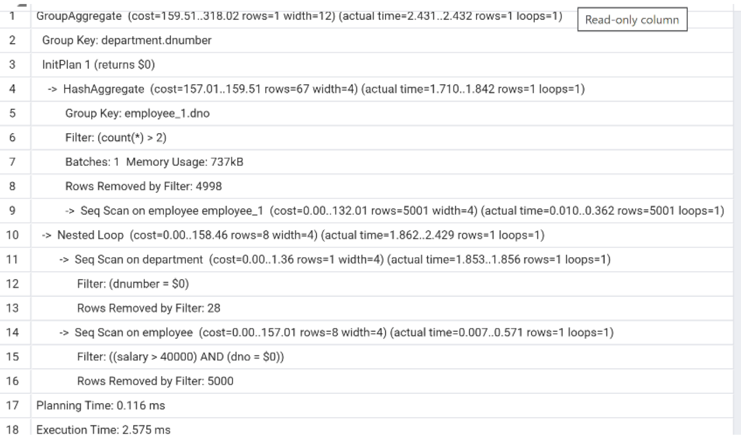
Scenario 1(No index)



```
GroupAggregate  (cost=159.51..318.02 rows=1 width=12) (actual time=2.696..2.697 rows=1 loops=1)
  Group Key: department.dnumber
  InitPlan 1 (returns $0)
    ->  HashAggregate  (cost=157.01..159.51 rows=67 width=4) (actual time=1.911..2.052 rows=1 loops=1)
          Group Key: employee_1.dno
          Filter: (count(*) > 2)
          Batches: 1  Memory Usage: 737kB
          Rows Removed by Filter: 4998
          ->  Seq Scan on employee employee_1  (cost=0.00..132.01 rows=5001 width=4) (actual time=0.009..0.486 rows=5001 loops=1)
  ->  Nested Loop  (cost=0.00..158.46 rows=8 width=4) (actual time=2.078..2.692 rows=1 loops=1)
        ->  Seq Scan on department  (cost=0.00..1.36 rows=1 width=4) (actual time=2.063..2.064 rows=1 loops=1)
              Filter: (dnumber = $0)
              Rows Removed by Filter: 28
        ->  Seq Scan on employee  (cost=0.00..157.01 rows=8 width=4) (actual time=0.012..0.624 rows=1 loops=1)
              Filter: ((salary > 40000) AND (dno = $0))
              Rows Removed by Filter: 5000
Planning Time: 0.403 ms
Execution Time: 2.820 ms
```

Scenario 2(B+ Tree index on Dnumber on department)

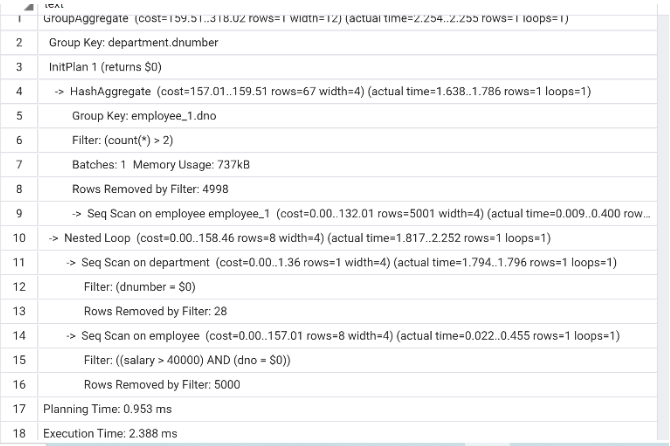
“In the nested query the dno is always compared so having a B+ index on it decreased the time taken to execute it”



```
GroupAggregate  (cost=159.51..318.02 rows=1 width=12) (actual time=2.431..2.432 rows=1 loops=1)
  Group Key: department.dnumber
  InitPlan 1 (returns $0)
    ->  HashAggregate  (cost=157.01..159.51 rows=67 width=4) (actual time=1.710..1.842 rows=1 loops=1)
          Group Key: employee_1.dno
          Filter: (count(*) > 2)
          Batches: 1  Memory Usage: 737kB
          Rows Removed by Filter: 4998
          ->  Seq Scan on employee employee_1  (cost=0.00..132.01 rows=5001 width=4) (actual time=0.010..0.362 rows=5001 loops=1)
  ->  Nested Loop  (cost=0.00..158.46 rows=8 width=4) (actual time=1.862..2.429 rows=1 loops=1)
        ->  Seq Scan on department  (cost=0.00..1.36 rows=1 width=4) (actual time=1.853..1.856 rows=1 loops=1)
              Filter: (dnumber = $0)
              Rows Removed by Filter: 28
        ->  Seq Scan on employee  (cost=0.00..157.01 rows=8 width=4) (actual time=0.007..0.571 rows=1 loops=1)
              Filter: ((salary > 40000) AND (dno = $0))
              Rows Removed by Filter: 5000
Planning Time: 0.116 ms
Execution Time: 2.575 ms
```

Scenario 3(Hash index on Salary in Employee)

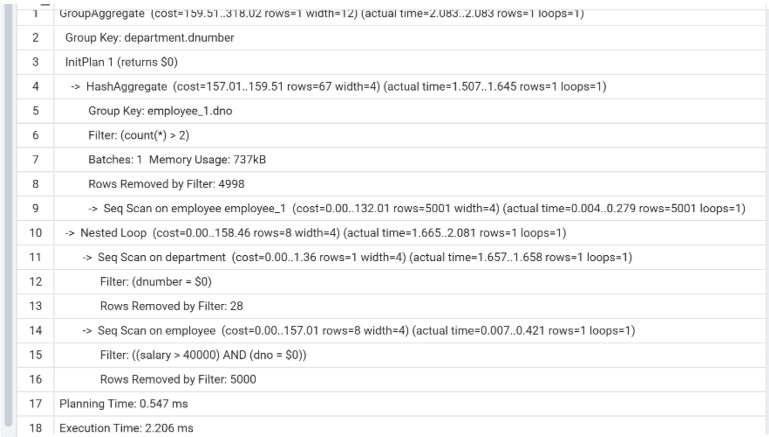
“There is a condition where the salary should always be greater than 40000 thus having a hash index on it will decrease the execution time”



```
text
GroupAggregate  (cost=159.51..318.02 rows=1 width=12) (actual time=2.254..2.255 rows=1 loops=1)
  Group Key: department.dnumber
  InitPlan 1 (returns $0)
    ->  HashAggregate  (cost=157.01..159.51 rows=67 width=4) (actual time=1.638..1.786 rows=1 loops=1)
          Group Key: employee_1.dno
          Filter: (count(*) > 2)
          Batches: 1  Memory Usage: 737kB
          Rows Removed by Filter: 4998
          ->  Seq Scan on employee employee_1  (cost=0.00..132.01 rows=5001 width=4) (actual time=0.009..0.400 row...
  ->  Nested Loop  (cost=0.00..158.46 rows=8 width=4) (actual time=1.817..2.252 rows=1 loops=1)
        ->  Seq Scan on department  (cost=0.00..1.36 rows=1 width=4) (actual time=1.794..1.796 rows=1 loops=1)
              Filter: (dnumber = $0)
              Rows Removed by Filter: 28
        ->  Seq Scan on employee  (cost=0.00..157.01 rows=8 width=4) (actual time=0.022..0.455 rows=1 loops=1)
              Filter: ((salary > 40000) AND (dno = $0))
              Rows Removed by Filter: 5000
Planning Time: 0.953 ms
Execution Time: 2.388 ms
```

Scenario 4(Mixed index [Hash on Salary,B+ on dnumber])

“Having both indexes resulted in the best performance in the execution time”

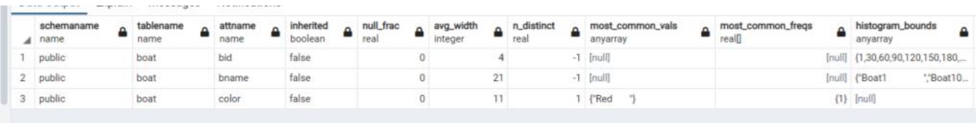


```
GroupAggregate  (cost=159.51..318.02 rows=1 width=12) (actual time=2.083..2.083 rows=1 loops=1)
  Group Key: department.dnumber
  InitPlan 1 (returns $0)
    ->  HashAggregate  (cost=157.01..159.51 rows=67 width=4) (actual time=1.507..1.645 rows=1 loops=1)
          Group Key: employee_1.dno
          Filter: (count(*) > 2)
          Batches: 1  Memory Usage: 737kB
          Rows Removed by Filter: 4998
          ->  Seq Scan on employee employee_1  (cost=0.00..132.01 rows=5001 width=4) (actual time=0.004..0.279 rows=5001 loops=1)
  ->  Nested Loop  (cost=0.00..158.46 rows=8 width=4) (actual time=1.665..2.081 rows=1 loops=1)
        ->  Seq Scan on department  (cost=0.00..1.36 rows=1 width=4) (actual time=1.657..1.658 rows=1 loops=1)
              Filter: (dnumber = $0)
              Rows Removed by Filter: 28
        ->  Seq Scan on employee  (cost=0.00..157.01 rows=8 width=4) (actual time=0.007..0.421 rows=1 loops=1)
              Filter: ((salary > 40000) AND (dno = $0))
              Rows Removed by Filter: 5000
Planning Time: 0.547 ms
Execution Time: 2.206 ms
```

Schema 3

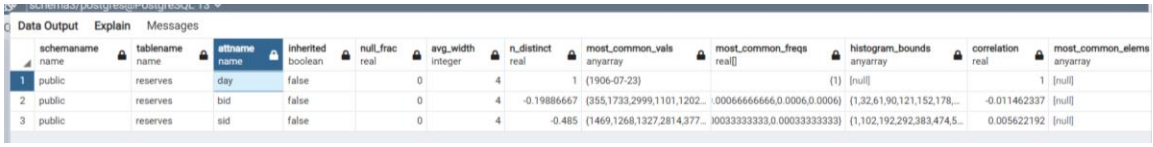
Query 7:

First of all, I analyzed both tables (sailors, reserves) to collect statistics to help us getting accurate estimates building the plan.



	schemaname	tablename	attname	inherited	null_frac	avg_width	n_distinct	most_common_vals	most_common_freqs	histogram_bounds
	name	name	name	boolean	real	integer	real	anyarray	real[]	anyarray
1	public	boat	bid	false	0	4	-1	[null]	[null]	{1,30,60,90,120,150,180,...
2	public	boat	bname	false	0	21	-1	[null]	[null]	{"Boat1 ","Boat10...
3	public	boat	color	false	0	11	1	{"Red "}	{1}	[null]

*“select * from pg_stats where tablename = ‘boat’ ”*



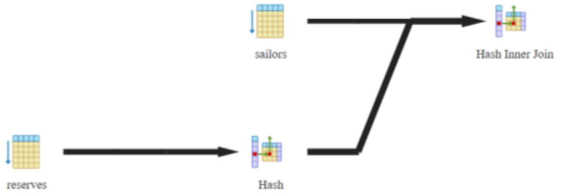
Data Output Explain Messages

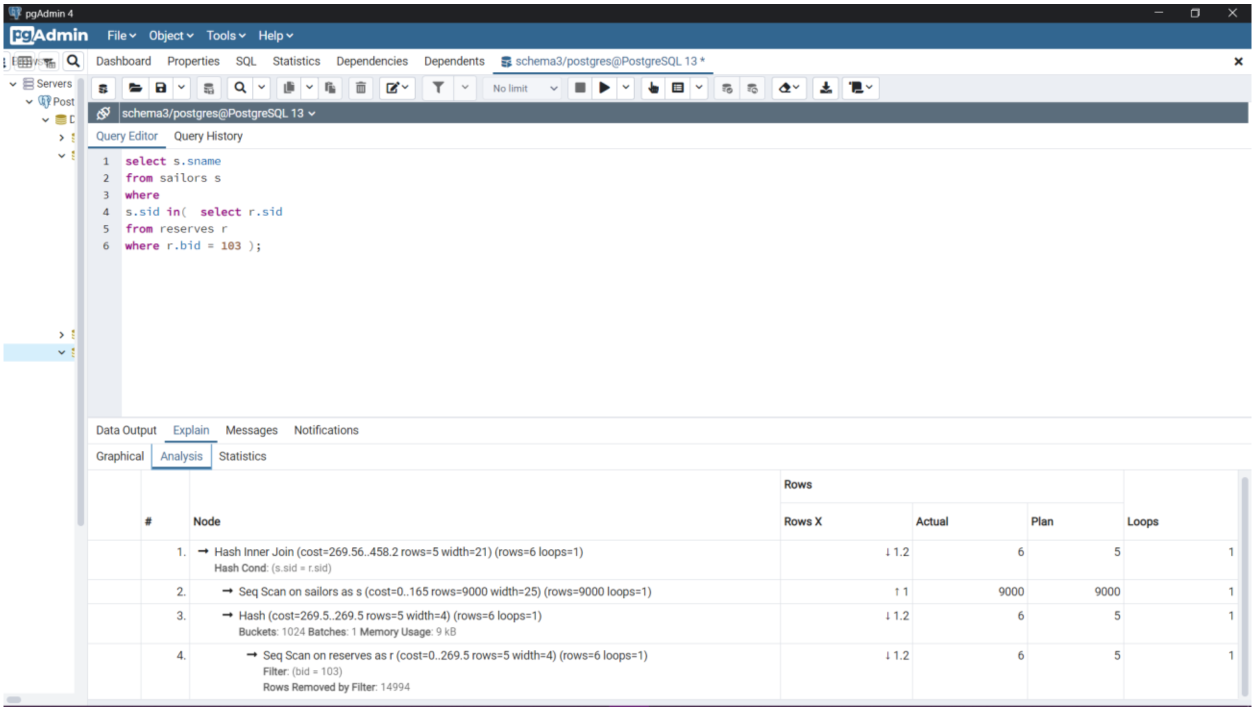
	schemaname	tablename	attname	inherited	null_frac	avg_width	n_distinct	most_common_vals	most_common_freqs	histogram_bounds	correlation	most_common_elems
	name	name	name	boolean	real	integer	real	anyarray	real[]	anyarray	real	anyarray
1	public	reserves	day	false	0	4	1	{1906-07-23}	{1}	[null]	1	[null]
2	public	reserves	bid	false	0	4	-0.19886667	{355,1733,2999,1101,1202...	.00066666666,0.0006,0.0006}	{1,32,61,90,121,152,178,...	-0.011462337	[null]
3	public	reserves	sid	false	0	4	-0.485	{1469,1268,1327,2814,377...	)0033333333,0.00033333333}	{1,102,192,292,383,474,5...	0.005622192	[null]

*“select * from pg_stats where tablename = ‘reserves’ ”*

Then I modified how PostgreSQL consider plans by disabling nested loops

“SET enable_nestloop=false”




pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents schema3/postgres@PostgreSQL 13 *
Servers
Post
No limit
schema3/postgres@PostgreSQL 13
Query Editor Query History
Data Output Explain Messages Notifications
Graphical Analysis Statistics

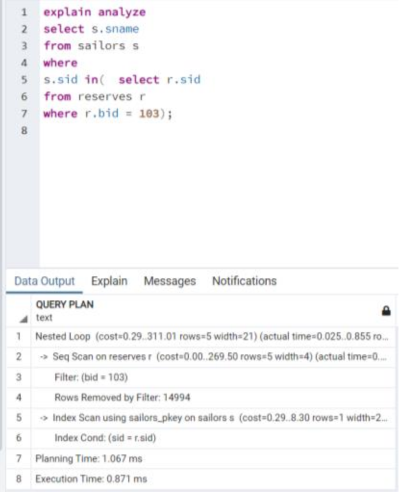
```
select s.sname
from sailors s
where
s.sid in(  select r.sid
from reserves r
where r.bid = 103 );
```

	#	Node	Rows			Loops
			Rows X	Actual	Plan	
	1.	→ Hash Inner Join (cost=269.56..458.2 rows=5 width=21) (rows=6 loops=1) Hash Cond: (s.sid = r.sid)	↓ 1.2	6	5	1
	2.	→ Seq Scan on sailors as s (cost=0..165 rows=9000 width=25) (rows=9000 loops=1)	↑ 1	9000	9000	1
	3.	→ Hash (cost=269.5..269.5 rows=5 width=4) (rows=6 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 9 kB	↓ 1.2	6	5	1
	4.	→ Seq Scan on reserves as r (cost=0..269.5 rows=5 width=4) (rows=6 loops=1) Filter: (bid = 103) Rows Removed by Filter: 14994	↓ 1.2	6	5	1

"Planning Time: 0.352 ms"

"Execution Time: 6.027 ms"

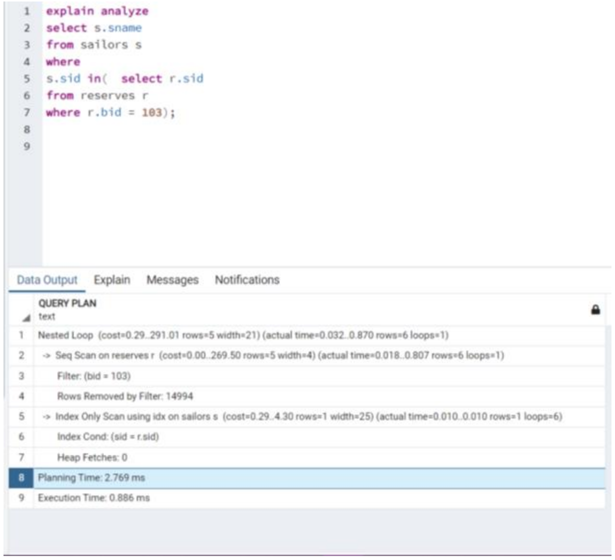
Scenario 1 (No index):



```sql
explain analyze
select s.sname
from sailors s
where
s.sid in(  select r.sid
from reserves r
where r.bid = 103);
```

```
QUERY PLAN
text
1  Nested Loop  (cost=0.29..311.01 rows=5 width=21) (actual time=0.025..0.855 ro...
2    -> Seq Scan on reserves r  (cost=0.00..269.50 rows=5 width=4) (actual time=0....
3         Filter: (bid = 103)
4         Rows Removed by Filter: 14994
5    -> Index Scan using sailors_pkey on sailors s  (cost=0.29..8.30 rows=1 width=2...
6         Index Cond: (sid = r.sid)
7  Planning Time: 1.067 ms
8  Execution Time: 0.871 ms
```

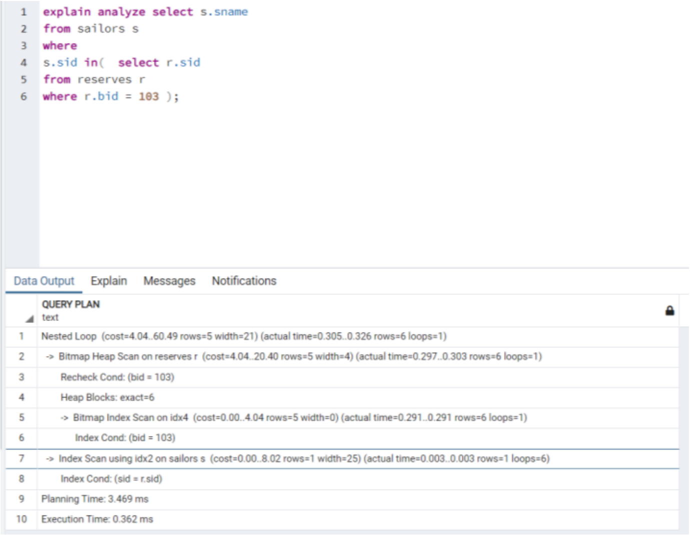
Scenario 2 (B+ tree index):



```sql
explain analyze
select s.sname
from sailors s
where
s.sid in(  select r.sid
from reserves r
where r.bid = 103);
```

```
QUERY PLAN
text
1  Nested Loop  (cost=0.29..291.01 rows=5 width=21) (actual time=0.032..0.870 rows=6 loops=1)
2    -> Seq Scan on reserves r  (cost=0.00..269.50 rows=5 width=4) (actual time=0.018..0.807 rows=6 loops=1)
3         Filter: (bid = 103)
4         Rows Removed by Filter: 14994
5    -> Index Only Scan using idx on sailors s  (cost=0.29..4.30 rows=1 width=25) (actual time=0.010..0.010 rows=1 loops=6)
6         Index Cond: (sid = r.sid)
7         Heap Fetches: 0
8  Planning Time: 2.769 ms
9  Execution Time: 0.886 ms
```

Scenario 3 (Hash index):

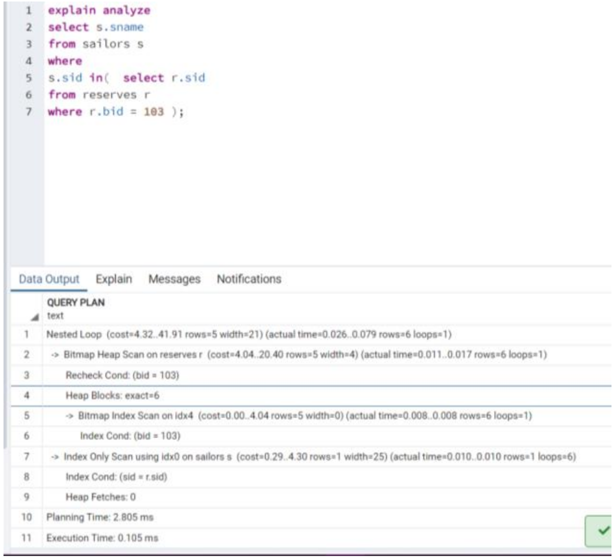


```sql
explain analyze select s.sname
from sailors s
where
s.sid in(  select r.sid
from reserves r
where r.bid = 103 );
```

```
   QUERY PLAN
   text
1  Nested Loop  (cost=4.04..60.49 rows=5 width=21) (actual time=0.305..0.326 rows=6 loops=1)
2    ->  Bitmap Heap Scan on reserves r  (cost=4.04..20.40 rows=5 width=4) (actual time=0.297..0.303 rows=6 loops=1)
3          Recheck Cond: (bid = 103)
4          Heap Blocks: exact=6
5          ->  Bitmap Index Scan on idx4  (cost=0.00..4.04 rows=5 width=0) (actual time=0.291..0.291 rows=6 loops=1)
6                Index Cond: (bid = 103)
7    ->  Index Scan using idx2 on sailors s  (cost=0.00..8.02 rows=1 width=25) (actual time=0.003..0.003 rows=1 loops=6)
8          Index Cond: (sid = r.sid)
9  Planning Time: 3.469 ms
10 Execution Time: 0.362 ms
```

Execution time improved drastically. As I created hash index on r.bid since hash indexes work really well with exact value queries in addition to an extra index on s.sid.

Scenario 4 (mixed indexes):



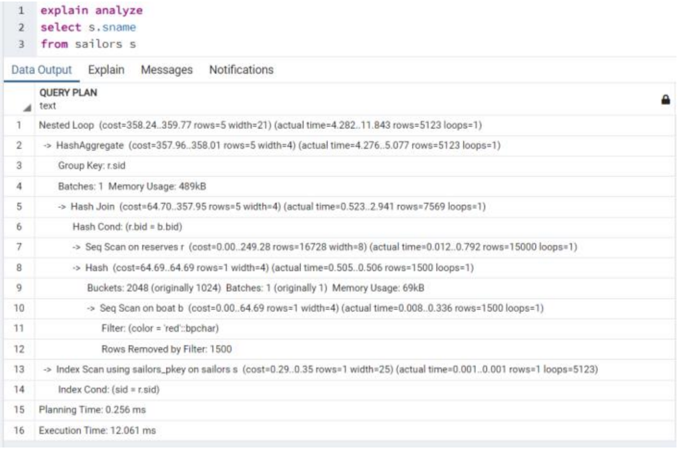
```sql
explain analyze
select s.sname
from sailors s
where
s.sid in(  select r.sid
from reserves r
where r.bid = 103 );
```

```
   QUERY PLAN
   text
1  Nested Loop  (cost=4.32..41.91 rows=5 width=21) (actual time=0.026..0.079 rows=6 loops=1)
2    ->  Bitmap Heap Scan on reserves r  (cost=4.04..20.40 rows=5 width=4) (actual time=0.011..0.017 rows=6 loops=1)
3          Recheck Cond: (bid = 103)
4          Heap Blocks: exact=6
5          ->  Bitmap Index Scan on idx4  (cost=0.00..4.04 rows=5 width=0) (actual time=0.008..0.008 rows=6 loops=1)
6                Index Cond: (bid = 103)
7    ->  Index Only Scan using idx0 on sailors s  (cost=0.29..4.30 rows=1 width=25) (actual time=0.010..0.010 rows=1 loops=6)
8          Index Cond: (sid = r.sid)
9          Heap Fetches: 0
10 Planning Time: 2.805 ms
11 Execution Time: 0.105 ms
```

Execution time improved drastically. As each index help in a specific way in optimizing the query.

Query 8:

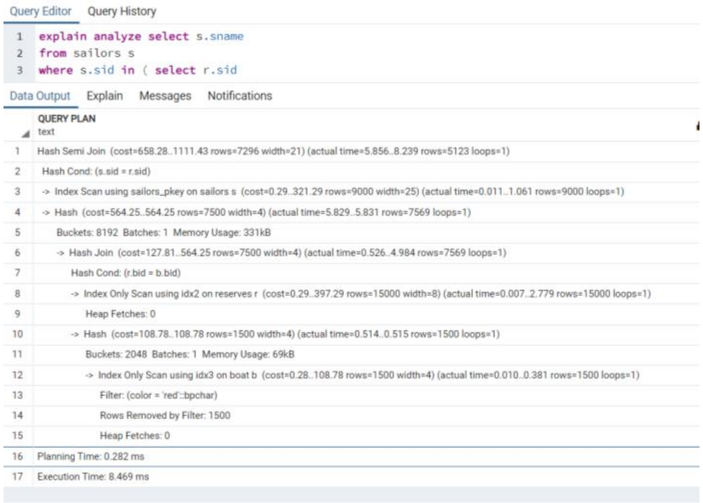
Scenario 1 (no index):



```sql
explain analyze
select s.sname
from sailors s
```

```
     QUERY PLAN
     text
1    Nested Loop  (cost=358.24..359.77 rows=5 width=21) (actual time=4.282..11.843 rows=5123 loops=1)
2      -> HashAggregate  (cost=357.96..358.01 rows=5 width=4) (actual time=4.276..5.077 rows=5123 loops=1)
3          Group Key: r.sid
4          Batches: 1  Memory Usage: 489kB
5          -> Hash Join  (cost=64.70..357.95 rows=5 width=4) (actual time=0.523..2.941 rows=7569 loops=1)
6              Hash Cond: (r.bid = b.bid)
7              -> Seq Scan on reserves r  (cost=0.00..249.28 rows=16728 width=8) (actual time=0.012..0.792 rows=15000 loops=1)
8              -> Hash  (cost=64.69..64.69 rows=1 width=4) (actual time=0.505..0.506 rows=1500 loops=1)
9                  Buckets: 2048 (originally 1024)  Batches: 1 (originally 1)  Memory Usage: 69kB
10                 -> Seq Scan on boat b  (cost=0.00..64.69 rows=1 width=4) (actual time=0.008..0.336 rows=1500 loops=1)
11                     Filter: (color = 'red'::bpchar)
12                     Rows Removed by Filter: 1500
13     -> Index Scan using sailors_pkey on sailors s  (cost=0.29..0.35 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=5123)
14         Index Cond: (sid = r.sid)
15   Planning Time: 0.256 ms
16   Execution Time: 12.061 ms
```

Scenario 2 (B+ tree index):

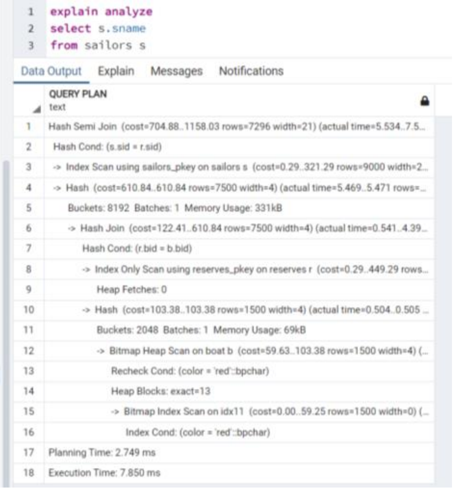


```sql
explain analyze select s.sname
from sailors s
where s.sid in ( select r.sid
```

```
     QUERY PLAN
     text
1    Hash Semi Join  (cost=658.28..1111.43 rows=7296 width=21) (actual time=5.856..8.239 rows=5123 loops=1)
2      Hash Cond: (s.sid = r.sid)
3      -> Index Scan using sailors_pkey on sailors s  (cost=0.29..321.29 rows=9000 width=25) (actual time=0.011..1.061 rows=9000 loops=1)
4      -> Hash  (cost=564.25..564.25 rows=7500 width=4) (actual time=5.829..5.831 rows=7569 loops=1)
5          Buckets: 8192  Batches: 1  Memory Usage: 331kB
6          -> Hash Join  (cost=127.81..564.25 rows=7500 width=4) (actual time=0.526..4.984 rows=7569 loops=1)
7              Hash Cond: (r.bid = b.bid)
8              -> Index Only Scan using idx2 on reserves r  (cost=0.29..397.29 rows=15000 width=8) (actual time=0.007..2.779 rows=15000 loops=1)
9                  Heap Fetches: 0
10             -> Hash  (cost=108.78..108.78 rows=1500 width=4) (actual time=0.514..0.515 rows=1500 loops=1)
11                 Buckets: 2048  Batches: 1  Memory Usage: 69kB
12                 -> Index Only Scan using idx3 on boat b  (cost=0.28..108.78 rows=1500 width=4) (actual time=0.010..0.381 rows=1500 loops=1)
13                     Filter: (color = 'red'::bpchar)
14                     Rows Removed by Filter: 1500
15                     Heap Fetches: 0
16   Planning Time: 0.282 ms
17   Execution Time: 8.469 ms
```

To force Postgres to use the created indexes I used “set enable_seqscan=false”, “set enable_indexscan =true”

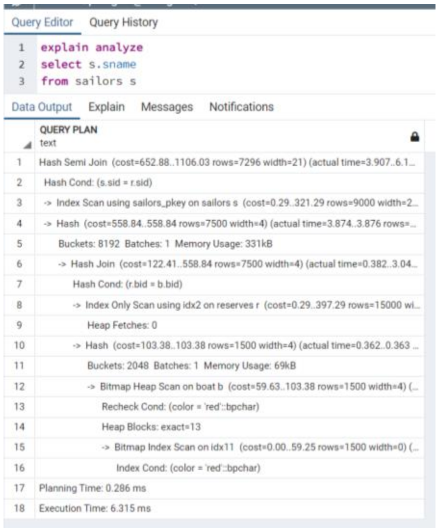
Scenario 3 (Hash index):



```sql
explain analyze
select s.sname
from sailors s
```

```
QUERY PLAN
text
Hash Semi Join  (cost=704.88..1158.03 rows=7296 width=21) (actual time=5.534..7.5...
  Hash Cond: (s.sid = r.sid)
  ->  Index Scan using sailors_pkey on sailors s  (cost=0.29..321.29 rows=9000 width=2...
  ->  Hash  (cost=610.84..610.84 rows=7500 width=4) (actual time=5.469..5.471 rows=...
        Buckets: 8192  Batches: 1  Memory Usage: 331kB
        ->  Hash Join  (cost=122.41..610.84 rows=7500 width=4) (actual time=0.541..4.39...
              Hash Cond: (r.bid = b.bid)
              ->  Index Only Scan using reserves_pkey on reserves r  (cost=0.29..449.29 rows...
                    Heap Fetches: 0
              ->  Hash  (cost=103.38..103.38 rows=1500 width=4) (actual time=0.504..0.505 ...
                    Buckets: 2048  Batches: 1  Memory Usage: 69kB
                    ->  Bitmap Heap Scan on boat b  (cost=59.63..103.38 rows=1500 width=4) (...
                          Recheck Cond: (color = 'red'::bpchar)
                          Heap Blocks: exact=13
                          ->  Bitmap Index Scan on idx11  (cost=0.00..59.25 rows=1500 width=0) (...
                                Index Cond: (color = 'red'::bpchar)
Planning Time: 2.749 ms
Execution Time: 7.850 ms
```

Scenario 4 (mixed indexes):

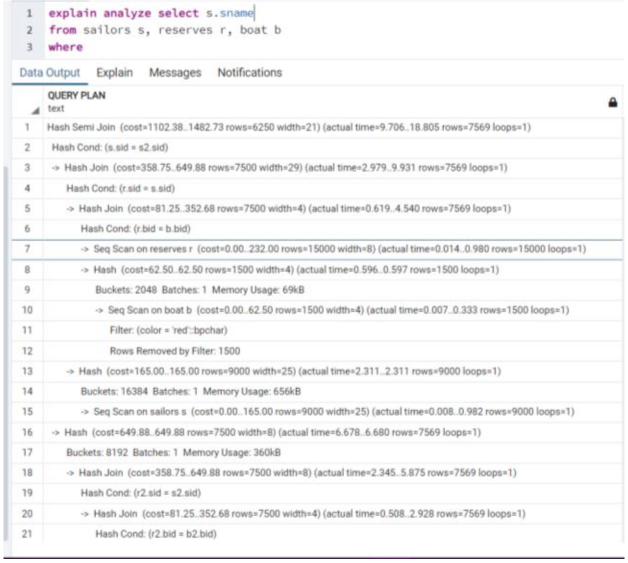


```sql
explain analyze
select s.sname
from sailors s
```

```
QUERY PLAN
text
Hash Semi Join  (cost=652.88..1106.03 rows=7296 width=21) (actual time=3.907..6.1...
  Hash Cond: (s.sid = r.sid)
  ->  Index Scan using sailors_pkey on sailors s  (cost=0.29..321.29 rows=9000 width=2...
  ->  Hash  (cost=558.84..558.84 rows=7500 width=4) (actual time=3.874..3.876 rows=...
        Buckets: 8192  Batches: 1  Memory Usage: 331kB
        ->  Hash Join  (cost=122.41..558.84 rows=7500 width=4) (actual time=0.382..3.04...
              Hash Cond: (r.bid = b.bid)
              ->  Index Only Scan using idx2 on reserves r  (cost=0.29..397.29 rows=15000 wi...
                    Heap Fetches: 0
              ->  Hash  (cost=103.38..103.38 rows=1500 width=4) (actual time=0.362..0.363 ...
                    Buckets: 2048  Batches: 1  Memory Usage: 69kB
                    ->  Bitmap Heap Scan on boat b  (cost=59.63..103.38 rows=1500 width=4) (...
                          Recheck Cond: (color = 'red'::bpchar)
                          Heap Blocks: exact=13
                          ->  Bitmap Index Scan on idx11  (cost=0.00..59.25 rows=1500 width=0) (...
                                Index Cond: (color = 'red'::bpchar)
Planning Time: 0.286 ms
Execution Time: 6.315 ms
```

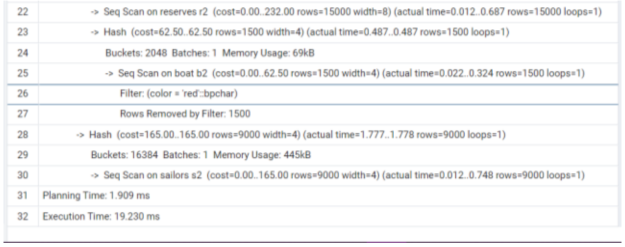
Query 9:

Scenario 1 (no index):



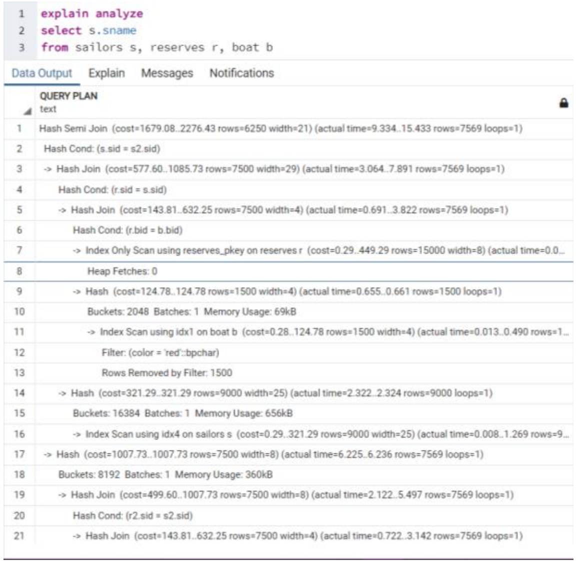
```
explain analyze select s.sname
from sailors s, reserves r, boat b
where
```

```
Hash Semi Join  (cost=1102.38..1482.73 rows=6250 width=21) (actual time=9.706..18.805 rows=7569 loops=1)
  Hash Cond: (s.sid = s2.sid)
  ->  Hash Join  (cost=358.75..649.88 rows=7500 width=29) (actual time=2.979..9.931 rows=7569 loops=1)
        Hash Cond: (r.sid = s.sid)
        ->  Hash Join  (cost=81.25..352.68 rows=7500 width=4) (actual time=0.619..4.540 rows=7569 loops=1)
              Hash Cond: (r.bid = b.bid)
              ->  Seq Scan on reserves r  (cost=0.00..232.00 rows=15000 width=8) (actual time=0.014..0.980 rows=15000 loops=1)
              ->  Hash  (cost=62.50..62.50 rows=1500 width=4) (actual time=0.596..0.597 rows=1500 loops=1)
                    Buckets: 2048  Batches: 1  Memory Usage: 69kB
                    ->  Seq Scan on boat b  (cost=0.00..62.50 rows=1500 width=4) (actual time=0.007..0.333 rows=1500 loops=1)
                          Filter: (color = 'red'::bpchar)
                          Rows Removed by Filter: 1500
        ->  Hash  (cost=165.00..165.00 rows=9000 width=25) (actual time=2.311..2.311 rows=9000 loops=1)
              Buckets: 16384  Batches: 1  Memory Usage: 656kB
              ->  Seq Scan on sailors s  (cost=0.00..165.00 rows=9000 width=25) (actual time=0.008..0.982 rows=9000 loops=1)
  ->  Hash  (cost=649.88..649.88 rows=7500 width=8) (actual time=6.678..6.680 rows=7569 loops=1)
        Buckets: 8192  Batches: 1  Memory Usage: 360kB
        ->  Hash Join  (cost=358.75..649.88 rows=7500 width=8) (actual time=2.345..5.875 rows=7569 loops=1)
              Hash Cond: (r2.sid = s2.sid)
              ->  Hash Join  (cost=81.25..352.68 rows=7500 width=4) (actual time=0.508..2.928 rows=7569 loops=1)
                    Hash Cond: (r2.bid = b2.bid)
```



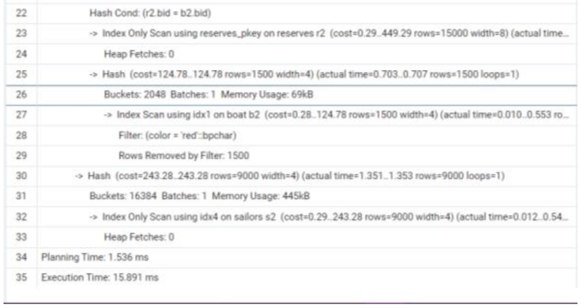
```
                    ->  Seq Scan on reserves r2  (cost=0.00..232.00 rows=15000 width=8) (actual time=0.012..0.687 rows=15000 loops=1)
                    ->  Hash  (cost=62.50..62.50 rows=1500 width=4) (actual time=0.487..0.487 rows=1500 loops=1)
                          Buckets: 2048  Batches: 1  Memory Usage: 69kB
                          ->  Seq Scan on boat b2  (cost=0.00..62.50 rows=1500 width=4) (actual time=0.022..0.324 rows=1500 loops=1)
                                Filter: (color = 'red'::bpchar)
                                Rows Removed by Filter: 1500
              ->  Hash  (cost=165.00..165.00 rows=9000 width=4) (actual time=1.777..1.778 rows=9000 loops=1)
                    Buckets: 16384  Batches: 1  Memory Usage: 445kB
                    ->  Seq Scan on sailors s2  (cost=0.00..165.00 rows=9000 width=4) (actual time=0.012..0.748 rows=9000 loops=1)
Planning Time: 1.909 ms
Execution Time: 19.230 ms
```

Scenario 2 (B+ tree index):



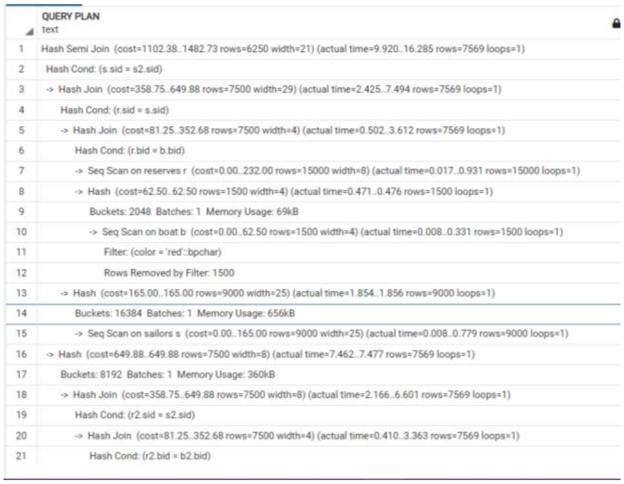
```sql
explain analyze
select s.sname
from sailors s, reserves r, boat b
```

```
   QUERY PLAN
   text
1  Hash Semi Join  (cost=1679.08..2276.43 rows=6250 width=21) (actual time=9.334..15.433 rows=7569 loops=1)
2    Hash Cond: (s.sid = s2.sid)
3    ->  Hash Join  (cost=577.60..1085.73 rows=7500 width=29) (actual time=3.064..7.891 rows=7569 loops=1)
4          Hash Cond: (r.sid = s.sid)
5          ->  Hash Join  (cost=143.81..632.25 rows=7500 width=4) (actual time=0.691..3.822 rows=7569 loops=1)
6                Hash Cond: (r.bid = b.bid)
7                ->  Index Only Scan using reserves_pkey on reserves r  (cost=0.29..449.29 rows=15000 width=8) (actual time=0.0...
8                      Heap Fetches: 0
9                ->  Hash  (cost=124.78..124.78 rows=1500 width=4) (actual time=0.655..0.661 rows=1500 loops=1)
10                     Buckets: 2048  Batches: 1  Memory Usage: 69kB
11                     ->  Index Scan using idx1 on boat b  (cost=0.28..124.78 rows=1500 width=4) (actual time=0.013..0.490 rows=1...
12                           Filter: (color = 'red'::bpchar)
13                           Rows Removed by Filter: 1500
14         ->  Hash  (cost=321.29..321.29 rows=9000 width=25) (actual time=2.322..2.324 rows=9000 loops=1)
15               Buckets: 16384  Batches: 1  Memory Usage: 656kB
16               ->  Index Scan using idx4 on sailors s  (cost=0.29..321.29 rows=9000 width=25) (actual time=0.008..1.269 rows=9...
17   ->  Hash  (cost=1007.73..1007.73 rows=7500 width=8) (actual time=6.225..6.236 rows=7569 loops=1)
18         Buckets: 8192  Batches: 1  Memory Usage: 360kB
19         ->  Hash Join  (cost=499.60..1007.73 rows=7500 width=8) (actual time=2.122..5.497 rows=7569 loops=1)
20               Hash Cond: (r2.sid = s2.sid)
21               ->  Hash Join  (cost=143.81..632.25 rows=7500 width=4) (actual time=0.722..3.142 rows=7569 loops=1)
```

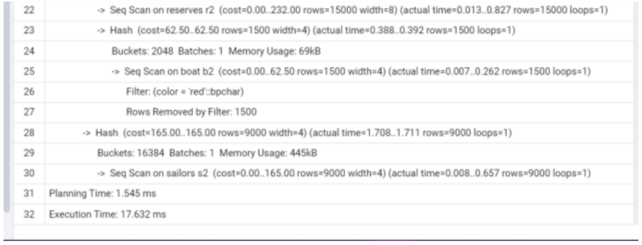


```
22                     Hash Cond: (r2.bid = b2.bid)
23                     ->  Index Only Scan using reserves_pkey on reserves r2  (cost=0.29..449.29 rows=15000 width=8) (actual time...
24                           Heap Fetches: 0
25                     ->  Hash  (cost=124.78..124.78 rows=1500 width=4) (actual time=0.703..0.707 rows=1500 loops=1)
26                           Buckets: 2048  Batches: 1  Memory Usage: 69kB
27                           ->  Index Scan using idx1 on boat b2  (cost=0.28..124.78 rows=1500 width=4) (actual time=0.010..0.553 ro...
28                                 Filter: (color = 'red'::bpchar)
29                                 Rows Removed by Filter: 1500
30               ->  Hash  (cost=243.28..243.28 rows=9000 width=4) (actual time=1.351..1.353 rows=9000 loops=1)
31                     Buckets: 16384  Batches: 1  Memory Usage: 445kB
32                     ->  Index Only Scan using idx4 on sailors s2  (cost=0.29..243.28 rows=9000 width=4) (actual time=0.012..0.54...
33                           Heap Fetches: 0
34 Planning Time: 1.536 ms
35 Execution Time: 15.891 ms
```

Scenario 3 (hash index):

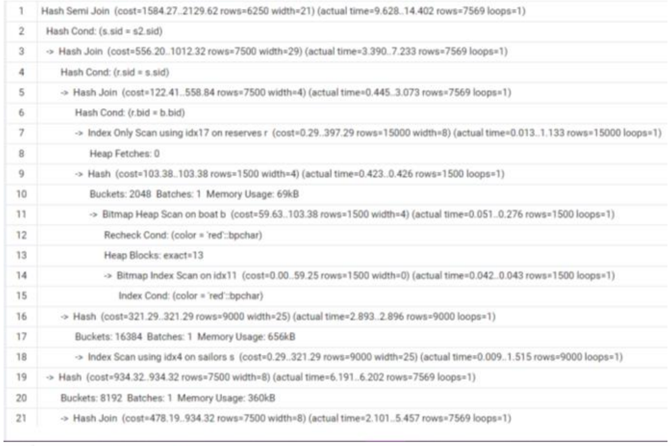


```
    QUERY PLAN
    text
1   Hash Semi Join  (cost=1102.38..1482.73 rows=6250 width=21) (actual time=9.920..16.285 rows=7569 loops=1)
2     Hash Cond: (s.sid = s2.sid)
3     ->  Hash Join  (cost=358.75..649.88 rows=7500 width=29) (actual time=2.425..7.494 rows=7569 loops=1)
4           Hash Cond: (r.sid = s.sid)
5           ->  Hash Join  (cost=81.25..352.68 rows=7500 width=4) (actual time=0.502..3.612 rows=7569 loops=1)
6                 Hash Cond: (r.bid = b.bid)
7                 ->  Seq Scan on reserves r  (cost=0.00..232.00 rows=15000 width=8) (actual time=0.017..0.931 rows=15000 loops=1)
8                 ->  Hash  (cost=62.50..62.50 rows=1500 width=4) (actual time=0.471..0.476 rows=1500 loops=1)
9                       Buckets: 2048  Batches: 1  Memory Usage: 69kB
10                      ->  Seq Scan on boat b  (cost=0.00..62.50 rows=1500 width=4) (actual time=0.008..0.331 rows=1500 loops=1)
11                            Filter: (color = 'red'::bpchar)
12                            Rows Removed by Filter: 1500
13          ->  Hash  (cost=165.00..165.00 rows=9000 width=25) (actual time=1.854..1.856 rows=9000 loops=1)
14                Buckets: 16384  Batches: 1  Memory Usage: 656kB
15                ->  Seq Scan on sailors s  (cost=0.00..165.00 rows=9000 width=25) (actual time=0.008..0.779 rows=9000 loops=1)
16    ->  Hash  (cost=649.88..649.88 rows=7500 width=8) (actual time=7.462..7.477 rows=7569 loops=1)
17          Buckets: 8192  Batches: 1  Memory Usage: 360kB
18          ->  Hash Join  (cost=358.75..649.88 rows=7500 width=8) (actual time=2.166..6.601 rows=7569 loops=1)
19                Hash Cond: (r2.sid = s2.sid)
20                ->  Hash Join  (cost=81.25..352.68 rows=7500 width=4) (actual time=0.410..3.363 rows=7569 loops=1)
21                      Hash Cond: (r2.bid = b2.bid)
```

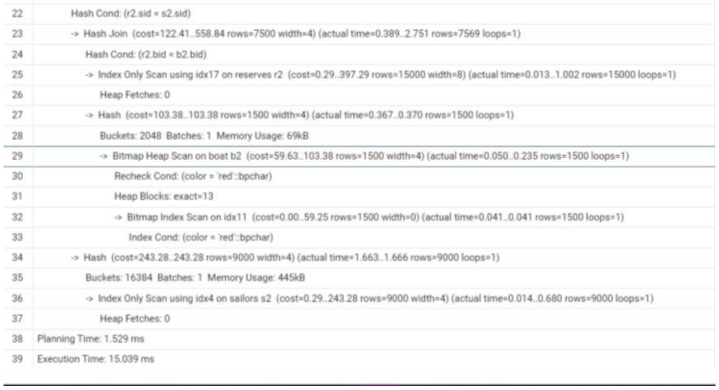


```
22                      ->  Seq Scan on reserves r2  (cost=0.00..232.00 rows=15000 width=8) (actual time=0.013..0.827 rows=15000 loops=1)
23                      ->  Hash  (cost=62.50..62.50 rows=1500 width=4) (actual time=0.388..0.392 rows=1500 loops=1)
24                            Buckets: 2048  Batches: 1  Memory Usage: 69kB
25                            ->  Seq Scan on boat b2  (cost=0.00..62.50 rows=1500 width=4) (actual time=0.007..0.262 rows=1500 loops=1)
26                                  Filter: (color = 'red'::bpchar)
27                                  Rows Removed by Filter: 1500
28                ->  Hash  (cost=165.00..165.00 rows=9000 width=4) (actual time=1.708..1.711 rows=9000 loops=1)
29                      Buckets: 16384  Batches: 1  Memory Usage: 445kB
30                      ->  Seq Scan on sailors s2  (cost=0.00..165.00 rows=9000 width=4) (actual time=0.008..0.657 rows=9000 loops=1)
31  Planning Time: 1.545 ms
32  Execution Time: 17.632 ms
```

Scenario 4 (mixed indexes):



```
Hash Semi Join  (cost=1584.27..2129.62 rows=6250 width=21) (actual time=9.628..14.402 rows=7569 loops=1)
  Hash Cond: (s.sid = s2.sid)
  ->  Hash Join  (cost=556.20..1012.32 rows=7500 width=29) (actual time=3.390..7.233 rows=7569 loops=1)
        Hash Cond: (r.sid = s.sid)
        ->  Hash Join  (cost=122.41..558.84 rows=7500 width=4) (actual time=0.445..3.073 rows=7569 loops=1)
              Hash Cond: (r.bid = b.bid)
              ->  Index Only Scan using idx17 on reserves r  (cost=0.29..397.29 rows=15000 width=8) (actual time=0.013..1.133 rows=15000 loops=1)
                    Heap Fetches: 0
              ->  Hash  (cost=103.38..103.38 rows=1500 width=4) (actual time=0.423..0.426 rows=1500 loops=1)
                    Buckets: 2048  Batches: 1  Memory Usage: 69kB
                    ->  Bitmap Heap Scan on boat b  (cost=59.63..103.38 rows=1500 width=4) (actual time=0.051..0.276 rows=1500 loops=1)
                          Recheck Cond: (color = 'red'::bpchar)
                          Heap Blocks: exact=13
                          ->  Bitmap Index Scan on idx11  (cost=0.00..59.25 rows=1500 width=0) (actual time=0.042..0.043 rows=1500 loops=1)
                                Index Cond: (color = 'red'::bpchar)
        ->  Hash  (cost=321.29..321.29 rows=9000 width=25) (actual time=2.893..2.896 rows=9000 loops=1)
              Buckets: 16384  Batches: 1  Memory Usage: 656kB
              ->  Index Scan using idx4 on sailors s  (cost=0.29..321.29 rows=9000 width=25) (actual time=0.009..1.515 rows=9000 loops=1)
  ->  Hash  (cost=934.32..934.32 rows=7500 width=8) (actual time=6.191..6.202 rows=7569 loops=1)
        Buckets: 8192  Batches: 1  Memory Usage: 360kB
        ->  Hash Join  (cost=478.19..934.32 rows=7500 width=8) (actual time=2.101..5.457 rows=7569 loops=1)
```



```
              Hash Cond: (r2.sid = s2.sid)
              ->  Hash Join  (cost=122.41..558.84 rows=7500 width=4) (actual time=0.389..2.751 rows=7569 loops=1)
                    Hash Cond: (r2.bid = b2.bid)
                    ->  Index Only Scan using idx17 on reserves r2  (cost=0.29..397.29 rows=15000 width=8) (actual time=0.013..1.002 rows=15000 loops=1)
                          Heap Fetches: 0
                    ->  Hash  (cost=103.38..103.38 rows=1500 width=4) (actual time=0.367..0.370 rows=1500 loops=1)
                          Buckets: 2048  Batches: 1  Memory Usage: 69kB
                          ->  Bitmap Heap Scan on boat b2  (cost=59.63..103.38 rows=1500 width=4) (actual time=0.050..0.235 rows=1500 loops=1)
                                Recheck Cond: (color = 'red'::bpchar)
                                Heap Blocks: exact=13
                                ->  Bitmap Index Scan on idx11  (cost=0.00..59.25 rows=1500 width=0) (actual time=0.041..0.041 rows=1500 loops=1)
                                      Index Cond: (color = 'red'::bpchar)
              ->  Hash  (cost=243.28..243.28 rows=9000 width=4) (actual time=1.663..1.666 rows=9000 loops=1)
                    Buckets: 16384  Batches: 1  Memory Usage: 445kB
                    ->  Index Only Scan using idx4 on sailors s2  (cost=0.29..243.28 rows=9000 width=4) (actual time=0.014..0.680 rows=9000 loops=1)
                          Heap Fetches: 0
Planning Time: 1.529 ms
Execution Time: 15.039 ms
```

Schema 4

Query 10 without index

Query plan:

```
"Nested Loop  (cost=228.29..231.72 rows=10 width=48) (actual time=0.162..0.163 rows=0 loops=1)"

"  ->  HashAggregate  (cost=228.00..228.10 rows=10 width=4) (actual time=0.162..0.163 rows=0 loops=1)"

"        Group Key: movie_cast.act_id"

"        Batches: 1  Memory Usage: 24kB"

"        ->  Nested Loop  (cost=0.29..227.98 rows=10 width=4) (actual time=0.162..0.162 rows=0 loops=1)"

"              ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual time=0.161..0.161 rows=0 loops=1)"

"                    Filter: (mov_title = 'movie1'::bpchar)"

"                    Rows Removed by Filter: 1000"

"              ->  Index Only Scan using movie_cast_pkey on movie_cast  (cost=0.29..195.38 rows=10 width=8) (never executed)"

"                    Index Cond: (mov_id = movie.mov_id)"

"                    Heap Fetches: 0"

"  ->  Index Scan using actor_pkey on actor  (cost=0.29..0.36 rows=1 width=48) (never executed)"

"        Index Cond: (act_id = movie_cast.act_id)"

"Planning Time: 0.322 ms"

"Execution Time: 0.202 ms"
```

RESULT CONCLUSION: Total time of 0.524 ms with no index.

Query 10 with B+ tree

Query plan:

```
"Nested Loop  (cost=228.29..231.72 rows=10 width=48) (actual time=0.113..0.114 rows=0 loops=1)"
"  ->  HashAggregate  (cost=228.00..228.10 rows=10 width=4) (actual time=0.113..0.113 rows=0 loops=1)"
"        Group Key: movie_cast.act_id"
"        Batches: 1  Memory Usage: 24kB"
"        ->  Nested Loop  (cost=0.29..227.98 rows=10 width=4) (actual time=0.112..0.113 rows=0 loops=1)"
"              ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual time=0.112..0.112 rows=0 loops=1)"
"                    Filter: (mov_title = 'movie1'::bpchar)"
"                    Rows Removed by Filter: 1000"
"              ->  Index Only Scan using movie_cast_pkey on movie_cast  (cost=0.29..195.38 rows=10 width=8) (never executed)"
"                    Index Cond: (mov_id = movie.mov_id)"
"                    Heap Fetches: 0"
"  ->  Index Scan using in1 on actor  (cost=0.29..0.36 rows=1 width=48) (never executed)"
"        Index Cond: (act_id = movie_cast.act_id)"
"Planning Time: 0.313 ms"
"Execution Time: 0.134 ms"
```

RESULT CONCLUSION: Total time of 0.447 ms very slightly made faster when using a B+ tree index.

Query 10 with hash index

Query plan:

```
"Nested Loop  (cost=66.48..67.47 rows=10 width=48) (actual time=0.129..0.130 rows=0 loops=1)"
"  ->  HashAggregate  (cost=66.48..66.58 rows=10 width=4) (actual time=0.129..0.129 rows=0 loops=1)"
"        Group Key: movie_cast.act_id"
"        Batches: 1  Memory Usage: 24kB"
"        ->  Nested Loop  (cost=4.08..66.45 rows=10 width=4) (actual time=0.128..0.128 rows=0 loops=1)"
"              ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual time=0.127..0.127 rows=0 loops=1)"
"                    Filter: (mov_title = 'movie1'::bpchar)"
"                    Rows Removed by Filter: 1000"
"              ->  Bitmap Heap Scan on movie_cast  (cost=4.08..33.85 rows=10 width=8) (never executed)"
"                    Recheck Cond: (mov_id = movie.mov_id)"
"                    ->  Bitmap Index Scan on in2  (cost=0.00..4.08 rows=10 width=0) (never executed)"
"                          Index Cond: (mov_id = movie.mov_id)"
"  ->  Index Scan using in1 on actor  (cost=0.00..0.09 rows=1 width=48) (never executed)"
"        Index Cond: (act_id = movie_cast.act_id)"
"Planning Time: 0.271 ms"
"Execution Time: 0.168 ms"
```

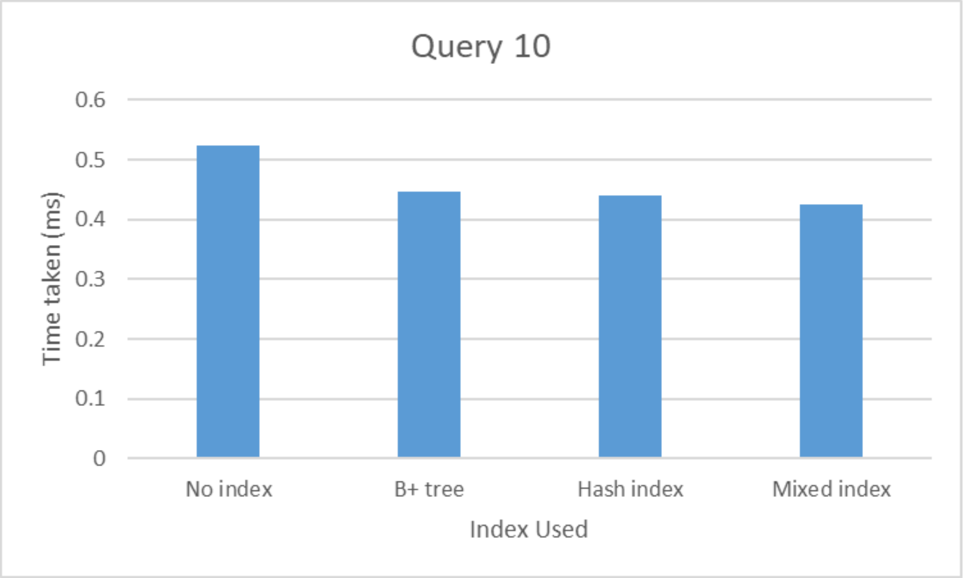
RESULT CONCLUSION: Total time 0.439 ms on par with the b+ tree speed.

Query 10 with mixed index

Query plan:

```
"Nested Loop  (cost=66.76..70.19 rows=10 width=48) (actual time=0.128..0.128 rows=0 loops=1)"

"  ->  HashAggregate  (cost=66.48..66.58 rows=10 width=4) (actual time=0.127..0.128 rows=0 loops=1)"

"        Group Key: movie_cast.act_id"

"        Batches: 1  Memory Usage: 24kB"

"        ->  Nested Loop  (cost=4.08..66.45 rows=10 width=4) (actual time=0.126..0.126 rows=0 loops=1)"

"              ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual time=0.126..0.126 rows=0 loops=1)"

"                    Filter: (mov_title = 'movie1'::bpchar)"

"                    Rows Removed by Filter: 1000"

"              ->  Bitmap Heap Scan on movie_cast  (cost=4.08..33.85 rows=10 width=8) (never executed)"

"                    Recheck Cond: (mov_id = movie.mov_id)"

"                    ->  Bitmap Index Scan on in2  (cost=0.00..4.08 rows=10 width=0) (never executed)"

"                          Index Cond: (mov_id = movie.mov_id)"

"  ->  Index Scan using in1 on actor  (cost=0.29..0.36 rows=1 width=48) (never executed)"

"        Index Cond: (act_id = movie_cast.act_id)"

"Planning Time: 0.266 ms"

"Execution Time: 0.158 ms"
```

RESULT CONCLUSION: Total time of 0.424 ms giving the best result set by mixing both hash based and b+ index for this query.



Query 11 without index

Query plan:

```
"Nested Loop  (cost=626.34..661.64 rows=100 width=42) (actual time=0.151..0.153 rows=0
loops=1)"
"  ->  HashAggregate  (cost=626.06..627.06 rows=100 width=4) (actual time=0.151..0.152
rows=0 loops=1)"
"        Group Key: movie_direction.dir_id"
"        Batches: 1  Memory Usage: 24kB"
"        ->  Hash Semi Join  (cost=453.46..625.81 rows=100 width=4) (actual time=0.150..0.151
rows=0 loops=1)"
"              Hash Cond: (movie_direction.mov_id = movie_cast.mov_id)"
"              ->  Seq Scan on movie_direction  (cost=0.00..144.99 rows=9999 width=8) (actual
time=0.010..0.010 rows=1 loops=1)"
"              ->  Hash  (cost=453.33..453.33 rows=10 width=4) (actual time=0.138..0.139 rows=0
loops=1)"
"                    Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                    ->  Hash Semi Join  (cost=242.99..453.33 rows=10 width=4) (actual
time=0.137..0.138 rows=0 loops=1)"
"                          Hash Cond: (movie_cast.role = movie_cast_1.role)"
"                          ->  Seq Scan on movie_cast  (cost=0.00..183.99 rows=9999 width=35) (actual
time=0.005..0.006 rows=1 loops=1)"
```

```
"                  ->  Hash  (cost=242.86..242.86 rows=10 width=31) (actual time=0.130..0.131
rows=0 loops=1)"
"                      Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                      ->  Hash Join  (cost=32.51..242.86 rows=10 width=31) (actual
time=0.130..0.131 rows=0 loops=1)"
"                          Hash Cond: (movie_cast_1.mov_id = movie.mov_id)"
"                          ->  Seq Scan on movie_cast movie_cast_1  (cost=0.00..183.99
rows=9999 width=35) (actual time=0.005..0.005 rows=1 loops=1)"
"                          ->  Hash  (cost=32.50..32.50 rows=1 width=4) (actual
time=0.125..0.125 rows=0 loops=1)"
"                              Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                              ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual
time=0.124..0.124 rows=0 loops=1)"
"                                  Filter: (mov_title = 'movie"
"2'::bpchar)"
"                                  Rows Removed by Filter: 1000"
"  ->  Index Scan using director_pkey on director  (cost=0.28..0.35 rows=1 width=46) (never
executed)"
"      Index Cond: (dir_id = movie_direction.dir_id)"
"Planning Time: 0.387 ms"
"Execution Time: 0.181 ms"
```

RESULT CONCLUSION: Total time 0.568 ms without an index.

Query 11 with B+ tree

Query plan:

```
"Nested Loop  (cost=450.21..485.52 rows=100 width=42) (actual time=0.129..0.131 rows=0
loops=1)"
"  ->  HashAggregate  (cost=449.94..450.94 rows=100 width=4) (actual time=0.129..0.130
rows=0 loops=1)"
"      Group Key: movie_direction.dir_id"
"      Batches: 1  Memory Usage: 24kB"
"      ->  Hash Semi Join  (cost=277.34..449.69 rows=100 width=4) (actual time=0.128..0.129
rows=0 loops=1)"
"          Hash Cond: (movie_direction.mov_id = movie_cast.mov_id)"
"          ->  Seq Scan on movie_direction  (cost=0.00..144.99 rows=9999 width=8) (actual
time=0.007..0.008 rows=1 loops=1)"
"          ->  Hash  (cost=277.21..277.21 rows=10 width=4) (actual time=0.119..0.119 rows=0
loops=1)"
"              Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                    ->  Hash Semi Join  (cost=66.86..277.21 rows=10 width=4) (actual
time=0.118..0.119 rows=0 loops=1)"
"                          Hash Cond: (movie_cast.role = movie_cast_1.role)"
"                          ->  Seq Scan on movie_cast  (cost=0.00..183.99 rows=9999 width=35) (actual
time=0.005..0.005 rows=1 loops=1)"
"                          ->  Hash  (cost=66.74..66.74 rows=10 width=31) (actual time=0.112..0.112
rows=0 loops=1)"
"                                Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                                ->  Nested Loop  (cost=4.36..66.74 rows=10 width=31) (actual
time=0.111..0.112 rows=0 loops=1)"
"                                      ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual
time=0.111..0.111 rows=0 loops=1)"
"                                            Filter: (mov_title = 'movie"
"2'::bpchar)"
"                                            Rows Removed by Filter: 1000"
"                                      ->  Bitmap Heap Scan on movie_cast movie_cast_1  (cost=4.36..34.14
rows=10 width=35) (never executed)"
"                                            Recheck Cond: (mov_id = movie.mov_id)"
"                                            ->  Bitmap Index Scan on in3  (cost=0.00..4.36 rows=10 width=0)
(never executed)"
"                                                  Index Cond: (mov_id = movie.mov_id)"
"  ->  Index Scan using in1 on director  (cost=0.28..0.35 rows=1 width=46) (never executed)"
"        Index Cond: (dir_id = movie_direction.dir_id)"
"Planning Time: 0.445 ms"
"Execution Time: 0.168 ms"
```

RESULT CONCLUSION: total time of 0.613 ms which is a significant increase from using no index.

Query 11 with hash index

Query plan:

```
"Nested Loop  (cost=449.65..457.86 rows=100 width=42) (actual time=0.174..0.176 rows=0
loops=1)"
"  ->  HashAggregate  (cost=449.65..450.65 rows=100 width=4) (actual time=0.174..0.175
rows=0 loops=1)"
"        Group Key: movie_direction.dir_id"
"        Batches: 1  Memory Usage: 24kB"
"        ->  Hash Semi Join  (cost=277.05..449.40 rows=100 width=4) (actual time=0.172..0.174
rows=0 loops=1)"
"              Hash Cond: (movie_direction.mov_id = movie_cast.mov_id)"
"              ->  Seq Scan on movie_direction  (cost=0.00..144.99 rows=9999 width=8) (actual
time=0.007..0.007 rows=1 loops=1)"
"              ->  Hash  (cost=276.93..276.93 rows=10 width=4) (actual time=0.163..0.164 rows=0
loops=1)"
"                    Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                    ->  Hash Semi Join  (cost=66.58..276.93 rows=10 width=4) (actual
time=0.163..0.164 rows=0 loops=1)"
"                          Hash Cond: (movie_cast.role = movie_cast_1.role)"
"                          ->  Seq Scan on movie_cast  (cost=0.00..183.99 rows=9999 width=35) (actual
time=0.005..0.005 rows=1 loops=1)"
"                          ->  Hash  (cost=66.45..66.45 rows=10 width=31) (actual time=0.157..0.157
rows=0 loops=1)"
"                                Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                                ->  Nested Loop  (cost=4.08..66.45 rows=10 width=31) (actual
time=0.156..0.157 rows=0 loops=1)"
"                                      ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual
time=0.156..0.156 rows=0 loops=1)"
"                                            Filter: (mov_title = 'movie"
"2'::bpchar)"
"                                            Rows Removed by Filter: 1000"
"                                      ->  Bitmap Heap Scan on movie_cast movie_cast_1  (cost=4.08..33.85
rows=10 width=35) (never executed)"
"                                            Recheck Cond: (mov_id = movie.mov_id)"
"                                            ->  Bitmap Index Scan on in3  (cost=0.00..4.08 rows=10 width=0)
(never executed)"
"                                                  Index Cond: (mov_id = movie.mov_id)"
"  ->  Index Scan using in1 on director  (cost=0.00..0.08 rows=1 width=46) (never executed)"
"        Index Cond: (dir_id = movie_direction.dir_id)"
"Planning Time: 0.476 ms"
"Execution Time: 0.209 ms"
```

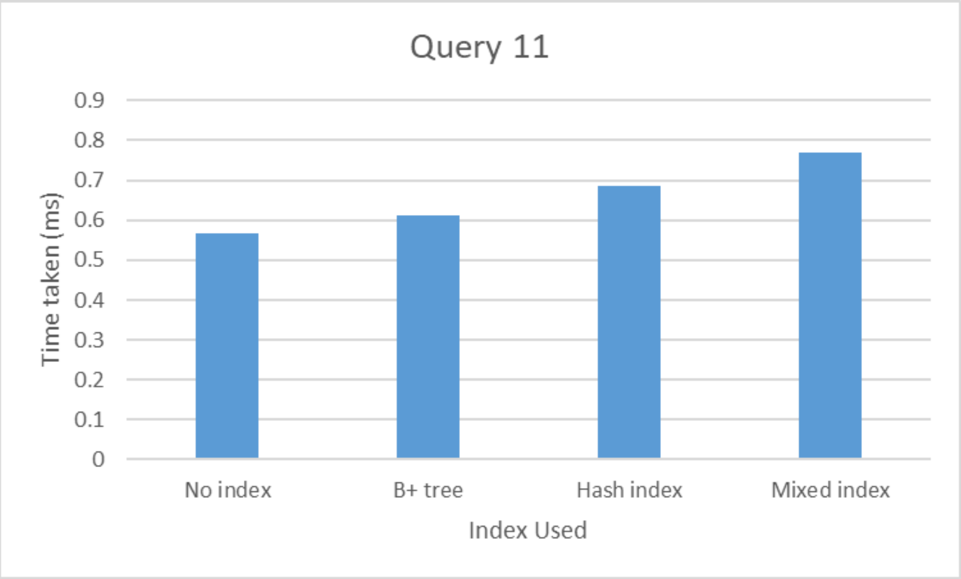
RESULT CONCLUSION: Total time of 0.685 ms an even greater decrease in performance from using b+ index.

Query 11 with mixed index

Query plan:

```
"Nested Loop  (cost=449.93..485.23 rows=100 width=42) (actual time=0.270..0.273 rows=0
loops=1)"
"  ->  HashAggregate  (cost=449.65..450.65 rows=100 width=4) (actual time=0.269..0.272
rows=0 loops=1)"
"        Group Key: movie_direction.dir_id"
"        Batches: 1  Memory Usage: 24kB"
"        ->  Hash Semi Join  (cost=277.05..449.40 rows=100 width=4) (actual time=0.267..0.270
rows=0 loops=1)"
"              Hash Cond: (movie_direction.mov_id = movie_cast.mov_id)"
"              ->  Seq Scan on movie_direction  (cost=0.00..144.99 rows=9999 width=8) (actual
time=0.012..0.012 rows=1 loops=1)"
"              ->  Hash  (cost=276.93..276.93 rows=10 width=4) (actual time=0.251..0.254 rows=0
loops=1)"
"                    Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                    ->  Hash Semi Join  (cost=66.58..276.93 rows=10 width=4) (actual
time=0.251..0.253 rows=0 loops=1)"
"                          Hash Cond: (movie_cast.role = movie_cast_1.role)"
"                          ->  Seq Scan on movie_cast  (cost=0.00..183.99 rows=9999 width=35) (actual
time=0.008..0.008 rows=1 loops=1)"
"                          ->  Hash  (cost=66.45..66.45 rows=10 width=31) (actual time=0.241..0.242
rows=0 loops=1)"
"                                Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                                ->  Nested Loop  (cost=4.08..66.45 rows=10 width=31) (actual
time=0.240..0.241 rows=0 loops=1)"
"                                      ->  Seq Scan on movie  (cost=0.00..32.50 rows=1 width=4) (actual
time=0.240..0.240 rows=0 loops=1)"
"                                            Filter: (mov_title = 'movie"
"2'::bpchar)"
"                                            Rows Removed by Filter: 1000"
"                                      ->  Bitmap Heap Scan on movie_cast movie_cast_1  (cost=4.08..33.85
rows=10 width=35) (never executed)"
"                                            Recheck Cond: (mov_id = movie.mov_id)"
"                                            ->  Bitmap Index Scan on in3  (cost=0.00..4.08 rows=10 width=0)
(never executed)"
"                                                  Index Cond: (mov_id = movie.mov_id)"
"  ->  Index Scan using in1 on director  (cost=0.28..0.35 rows=1 width=46) (never executed)"
"        Index Cond: (dir_id = movie_direction.dir_id)"
"Planning Time: 0.446 ms"
"Execution Time: 0.323 ms"
```

RESULT CONCLUSION: Total time 0.769 ms which shows that using mixed index provides a significant decrease in performance on this partical query.



Query 12 without index

Query plan :

```
"Index Scan using movie_pkey on movie  (cost=53.65..61.67 rows=1 width=51) (actual time=0.182..0.183 rows=0 loops=1)"
"  Index Cond: (mov_id = $1)"
"  InitPlan 2 (returns $1)"
"    ->  Index Only Scan using movie_direction_pkey on movie_direction  (cost=49.28..53.37 rows=5 width=4) (actual time=0.180..0.180 rows=0 loops=1)"
"          Index Cond: (dir_id = $0)"
"          Heap Fetches: 0"
"          InitPlan 1 (returns $0)"
"            ->  Seq Scan on director  (cost=0.00..49.00 rows=1 width=4) (actual time=0.177..0.177 rows=0 loops=1)"
"                  Filter: ((dir_fname = 'actor1'::bpchar) AND (dir_lname = 'actor1'::bpchar))"
"                  Rows Removed by Filter: 2000"
"Planning Time: 0.110 ms"
"Execution Time: 0.200 ms"
```

RESULT CONCLUSION: Total time of 0.310 ms without an index.

Query 12 with b+ tree

Query plan :

```
"Index Scan using in1 on movie  (cost=53.65..61.67 rows=1 width=51) (actual
time=0.185..0.186 rows=0 loops=1)"
"  Index Cond: (mov_id = $1)"
"  InitPlan 2 (returns $1)"
"    ->  Index Only Scan using movie_direction_pkey on movie_direction  (cost=49.28..53.37
rows=5 width=4) (actual time=0.183..0.184 rows=0 loops=1)"
"          Index Cond: (dir_id = $0)"
"          Heap Fetches: 0"
"          InitPlan 1 (returns $0)"
"            ->  Seq Scan on director  (cost=0.00..49.00 rows=1 width=4) (actual time=0.180..0.180
rows=0 loops=1)"
"                  Filter: ((dir_fname = 'actor1'::bpchar) AND (dir_lname = 'actor1'::bpchar))"
"                  Rows Removed by Filter: 2000"
"Planning Time: 0.125 ms"
"Execution Time: 0.203 ms"
```

RESULT CONCLUSION: Total time of 0.328 ms not a significant change from using no index.

Query 12 with hash index

Query plan :

```
"Index Scan using in1 on movie  (cost=53.37..61.39 rows=1 width=51) (actual
time=0.220..0.221 rows=0 loops=1)"
"  Index Cond: (mov_id = $1)"
"  InitPlan 2 (returns $1)"
"    ->  Index Only Scan using movie_direction_pkey on movie_direction  (cost=49.28..53.37
rows=5 width=4) (actual time=0.218..0.219 rows=0 loops=1)"
"          Index Cond: (dir_id = $0)"
"          Heap Fetches: 0"
"          InitPlan 1 (returns $0)"
"            ->  Seq Scan on director  (cost=0.00..49.00 rows=1 width=4) (actual time=0.199..0.199
rows=0 loops=1)"
"                  Filter: ((dir_fname = 'actor1'::bpchar) AND (dir_lname = 'actor1'::bpchar))"
"                  Rows Removed by Filter: 2000"
"Planning Time: 0.143 ms"
"Execution Time: 0.259 ms"
```

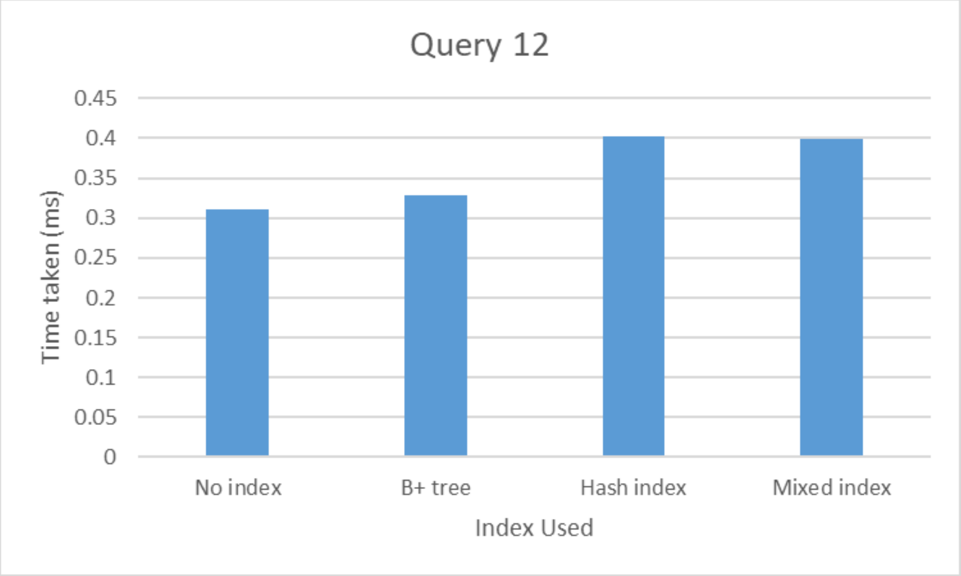
RESULT CONCLUSION: Total time of 0.402 ms a significant performance decrease using a hash index.

Query 12 with mixed index

Query plan :

```
"Index Scan using in1 on movie  (cost=53.37..61.39 rows=1 width=51) (actual
time=0.243..0.244 rows=0 loops=1)"
"  Index Cond: (mov_id = $1)"
"  InitPlan 2 (returns $1)"
"    ->  Index Only Scan using movie_direction_pkey on movie_direction  (cost=49.28..53.37
rows=5 width=4) (actual time=0.241..0.241 rows=0 loops=1)"
"          Index Cond: (dir_id = $0)"
"          Heap Fetches: 0"
"          InitPlan 1 (returns $0)"
"            ->  Seq Scan on director  (cost=0.00..49.00 rows=1 width=4) (actual time=0.236..0.236
rows=0 loops=1)"
"                  Filter: ((dir_fname = 'actor1'::bpchar) AND (dir_lname = 'actor1'::bpchar))"
"                  Rows Removed by Filter: 2000"
"Planning Time: 0.116 ms"
"Execution Time: 0.283 ms"
```

RESULT CONCLUSION: Total time 0.399 ms using mixed index gave similar results to using a hash index both being lower performance wise from no index or
b+ tree index.


Query 12
Time taken (ms)
No index
B+ tree
Hash index
Mixed index
Index Used