

Query 9

- Find the names of sailors who have reserved both a red and a green boat.

Original Query

```

select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red'
and
s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
and
r2.bid = b2.bid
and
b2.color = 'red');

```

Result Set

- 177 Rows

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Activities, pgAdmin 4, File, Object, Tools, Help.
- Servers:** Local Database Server, postgres.
- Databases:** postgres, schema1, schema2, schema3.
- Current Schema:** schema3/postgres@postgres*
- Query Editor:**
 - Text area containing the original SQL query.
 - Execution status: May 17 12:04 AM • pgAdmin 4
 - Buttons: Execute/Refresh (highlighted), F5, Stop, Run, Save, Copy, Paste, Find, Replace, Undo, Redo, Help.
- Results Grid:**

count	bignum
1	177
- Message Bar:** Total rows: 1 of 1 | Query complete 00:00:00.107 | Successfully run. Total query runtime: 51 msec. 582 rows affected. | Ln 170, Col 1

Report

1. given query without an index,

May 17 12:08 AM ● pgAdmin 4

```

15
16 select count(sid)
17 from sailors;
18
19 select count(*)
20 from boat;
21
22 select count(*)
23 from reserves;
24
25 delete from Reserves;
26
27 delete from sailors;
28
29 delete from Boat;
30
31
32 --- Sailors
33 -- see what indexes are created for that table
34 select *
35 from pg_indexes
36 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
37
38 -- see constraint names
39 SELECT con.*
40 FROM pg_catalog.pg_constraint con
41 INNER JOIN pg_catalog.pg_class rel
42

```

Total rows: 0 of 0 Query complete 00:00:00.117 Jun 8 11:18 PM ● pgAdmin 4

Successfully run. Total query runtime: 51 msec. 582 rows affected.

Jun 8 11:18 PM ● pgAdmin 4

```

280
281
282 -- Query 9 (STATISTICS)
283 set enable_hashagg = off;
284 set enable_hashjoin = off;
285 explain analyze select s.sname
286 from sailors s, reserves r, boat b
287 where
288 s.sid = r.sid
289 and
290 r.bid = b.bid
291 and
292 b.color = 'red'
293 and
294 s.sid in ( select s2.sid
295 from sailors s2, boat b2, reserves r2
296 where s2.sid = r2.sid
297 and
298 r2.bid = b2.bid
299 and
300 b2.color = 'green');
301
302
303 -- Query 9 optimization (STATISTICS)
304 -- Find the names of sailors who have reserved both a red and a green boat.
305
306
307
308 select *
309 from pg_indexes
310 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
311
312
313 explain analyze select s.sname
314 from sailors s
315 where exists
316 (
317 select rTotal.sid
318 from (select r1.sid
319 from
320 (select r.sid

```

Total rows: 1 of 1 Query complete 00:00:00.090 Ln 301, Col 1

The diagram illustrates the execution plan for the provided SQL query. It shows four parallel execution paths. Each path starts with a 'Sort' node, followed by a 'Merge Inner Join' node. The first two paths (top-left and top-right) involve the 'reserves' table, while the last two (bottom-left and bottom-right) involve the 'sailors' table. The 'boat' table is also mentioned in the plan. The nodes are interconnected by lines representing data flow between the tables and their respective sorting and joining stages.

```

set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
    s.sid = r.sid
    and
    r.bid = b.bid
    and
    b.color = 'red'
    and
    s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
    and
    r2.bid = b2.bid
    and
    b2.color = 'green');

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
    
```

Total rows: 46 of 46 Query complete 00:00:00.223 Ln 283, Col 1

```

set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
    s.sid = r.sid
    and
    r.bid = b.bid
    and
    b.color = 'red'
    and
    s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
    and
    r2.bid = b2.bid
    and
    b2.color = 'green');

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
    
```

Total rows: 46 of 46 Query complete 00:00:00.223 Ln 283, Col 1

Explanation :

- Metrics :

Execution Time : 64.569 ms Total Expected Cost : 11368.59

- Flags Hashjoin and HashAgg here where disabled for future after many trials and errors , I've discovered the best way to show the difference in terms of the cost and to beat the Postgres Query Optimizer Algorithm to be able to show indices effect and cost differences .

2. given query with B+ trees indices only,

pgAdmin 4 - Jun 9 12:37 AM • pgAdmin 4

```

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
        from reserves r
        where exists
        (select bid
        from boat b
        where color = 'green' and r.bid = b.bid )
        ) as r1
    inner join
        (select r.sid
        from reserves r
        where exists
        (select bid
        from boat b
        where color = 'red' and r.bid = b.bid )
        ) as r2
    on r2.sid = r1.sid ) as rTotal
    where rTotal.sid=s.sid
);

Total rows: 4 of 4   Query complete 00:00:00.224
Ln 308, Col 1

```

pgAdmin 4 - Jun 9 12:15 AM • pgAdmin 4

```

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red';

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
        from reserves r
        where exists
        (select bid
        from boat b
        where color = 'red' and r.bid = b.bid )
        ) as r1
    inner join
        (select r.sid
        from reserves r
        where exists
        (select bid
        from boat b
        where color = 'green' and r.bid = b.bid )
        ) as r2
    on r2.sid = r1.sid ) as rTotal
    where rTotal.sid=s.sid
);

Total rows: 29 of 29   Query complete 00:00:00.153
Ln 300, Col 1

```

The screenshot shows the pgAdmin 4 interface with the Explain Graph tab selected. The query being explained is:

```

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
    s.sid = r.sid
    and
    r.bid = b.bid
    and
    b.color = 'red';
-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat
select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s
where exists
(
    ...
);
  
```

The Explain Graph shows the following execution plan:

- Input Tables:** boat, b_reservesbid, b_sailorssid.
- Operations:**
 - Nested Loop Inner Join between boat and b_reservesbid.
 - Sort operation on b_reservesbid.
 - Merge Inner Join between b_reservesbid and b_sailorssid.
 - Nested Loop Inner Join between boat and b_sailorssid.
 - Sort operation on b_sailorssid.
 - Merge Semi Join between the two results of the previous steps.
- Final Output:** The result of the Merge Semi Join.

The status bar at the bottom right indicates: Successfully run. Total query runtime: 176 msec. 1 rows affected. Ln 285, Col 16.

Explanation :

- Metrics :

Execution Time : 7.925 ms	Total Expected Cost : 5557.12
----- ----- -----	

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost .
- The Query Planner used the B-tree index because B-Tree is O(Log n) performance with Exact Values
- Here it showed the improvement due to for every condition of Joining in the Query the Merge Join used index scan using ZigZag algorithm .

3. given query with hash indices only,

pgAdmin 4

Jun 9 12:48 AM •

Servers Local PostgreSQL

Data Query History

```

inner join
  (select r.sid
   from reserves r
   where exists
     (select bid
      from boat b
      where color = 'red' and r.bid = b.bid )
    ) as rTotal
   on r2.sid = r1.sid ) as rTotal
  where rTotal.sid=s.sid

CREATE INDEX b_sailorsSID ON sailors USING HASH(sid);
CREATE INDEX b_reservesSID ON reserves USING HASH(sid);
CREATE INDEX b_reservesRID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);
CREATE INDEX b_boat2 ON boat USING HASH(color);

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

```

Total rows: 5 of 5 Query complete 00:00:00.050

Ln 277, Col 1

pgAdmin 4

Jun 9 1:02 AM •

Servers Local PostgreSQL

Data Query History

```

CREATE INDEX b_reservesSID ON reserves USING HASH(sid);
CREATE INDEX b_reservesRID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);
CREATE INDEX b_boat2 ON boat USING HASH(color);

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red'
and
s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
and
r2.bid = b2.bid
and
b2.color = 'green');

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

```

QUERY PLAN

text
1 Nested Loop Semi Join (cost=0.00..3052.70 rows=1309 width=21) (actual time=3.521..12.593 rows=177 loops=1)
2 Join Filter: (s.sid = s2.sid)
3 -> Nested Loop (cost=0.00..2008.40 rows=4982 width=29) (actual time=0.029..3.816 rows=1136 loops=1)
4 -> Nested Loop (cost=0.00..1743.29 rows=4982 width=4) (actual time=0.024..1.734 rows=1136 loops=1)
5 -> Seq Scan on boat b (cost=0.00..62.50 rows=427 width=4) (actual time=0.014..0.452 rows=427 loops=1)
6 Filter: (color = 'red':bpchar)
7 Rows Removed by Filter: 2573
8 -> Index Scan using b_reservesRID on reserves r (cost=0.00..3.66 rows=28 width=8) (actual time=0.001..0.002 rows=3 loops=427)
9 Index Cond: (bid = b.bid)
10 -> Index Scan using b_sailorsSID on sailors s (cost=0.00..0.04 rows=1 width=25) (actual time=0.001..0.001 rows=1 loops=1136)
11 Index Cond: (sid = r.sid)
12 -> Nested Loop (cost=0.00..0.20 rows=1 width=8) (actual time=0.007..0.007 rows=0 loops=1136)
13 Join Filter: (r2.sid = s2.sid)
14 -> Nested Loop (cost=0.00..0.14 rows=1 width=4) (actual time=0.007..0.007 rows=0 loops=1136)
15 -> Index Scan using b_reservesRID on reserves r2 (cost=0.00..0.07 rows=2 width=8) (actual time=0.001..0.002 rows=3 loops=1136)
16 Index Cond: (sid = r2.sid)
17 -> Index Scan using b_boat1 on boat b2 (cost=0.00..0.02 rows=1 width=4) (actual time=0.001..0.001 rows=0 loops=3199)
18 Index Cond: (bid = r2.bid)
19 Filter: (color = 'green':bpchar)
20 Rows Removed by Filter: 1
21 -> Index Scan using b_sailorsSID on sailors s2 (cost=0.00..0.04 rows=1 width=4) (actual time=0.001..0.001 rows=1 loops=177)
22 Index Cond: (sid = r.sid)
23 Planning Time: 0.589 ms
24 Execution Time: 12.646 ms

Total rows: 24 of 24 Query complete 00:00:00.072

Ln 286, Col 1

The screenshot shows the pgAdmin 4 interface with a query editor and an explain plan viewer. The query editor contains a complex SQL script for creating indexes and performing a multi-table join. The explain plan on the right details the execution flow, including nested loop joins and the use of hash indexes for specific joins.

```

CREATE INDEX b_reservesID ON reserves USING HASH(bid);
CREATE INDEX b_reservesBID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);
CREATE INDEX b_boat2 ON boat USING HASH(color);

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red'
and
s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
and
r2.bid = b2.bid
and
b2.color = 'green');
-- Find the names of sailors who have reserved both a red and a green boat.

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

```

Explanation :

- Metrics :

Execution Time : 12.6 ms Total Expected Cost : 3052.12
--

- The Hash helped in the performance it decreased the Execution Time (but in the execution time processor was overwhelmed) and Expected Cost .
- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries.
- Here it showed the improvement due to for every condition of Joining in the Query the Nested Loop Join using index scan using Hash based algorithm which approximatlly maded to be O(n) performance.
- The Where clause condition on the color used Hash too.

4. given query with BRIN indices only,

pgAdmin 4 - Jun 9 2:29 AM

schema3/postgres@postgres

Data output Messages Explain Notifications

indexdef	text
1	CREATE INDEX b_sailorsSID ON sailors USING BRIN(sid);
2	CREATE INDEX b_reservesSID ON reserves USING BRIN(sid);
3	CREATE INDEX b_reservesBID ON reserves USING BRIN(bid);
4	CREATE INDEX b_boat ON boat USING BRIN(bid,color);

```

267 CREATE INDEX b_sailorsSID ON sailors USING BRIN(sid);
268 CREATE INDEX b_reservesSID ON reserves USING BRIN(sid );
269 CREATE INDEX b_reservesBID ON reserves USING BRIN(bid );
270 CREATE INDEX b_boat ON boat USING BRIN(bid,color );

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

282 -- Query 9 (STATISTICS)
283 set enable_hashagg = off;
284 set enable_hashjoin = off;
285 explain analyze select s.sname
286   from sailors s, reserves r, boat b
287 where
288   s.sid = r.sid
289   and
290   r.bid = b.bid
291   and
292   b.color = 'red'
293   and
294   s.sid in ( select s2.sid
295     from sailors s2, boat b2, reserves r2
296   where s2.sid = r2.sid
297   and
298   r2.bid = b2.bid
299   and
300   b2.color = 'green';
301
302
303 -- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.
Total rows: 4 of 4 Query complete 00:00:00.0088

```

Ln 273, Col 1

pgAdmin 4 - Jun 9 2:48 AM

schema3/postgres@postgres

Data output Messages Explain Notifications

QUERY PLAN

text
1 Nested Loop (cost=210266133.97..2401050935.23 rows=1309 width=21) (actual time=4472.120..4675.994 rows=177 loops=1)
2 Join Filter: (r.sid = s.sid)
3 -> Merge Join (cost=2102438133.93..2102438196.90 rows=1309 width=12) (actual time=4472.077..4472.526 rows=177 loops=1)
4 Merge Cond: (s.sid = r.sid)
5 -> Sort (cost=2376998.97..2377011.43 rows=4982 width=4) (actual time=1143.068..1143.151 rows=1136 loops=1)
6 Sort Key: r.sid
7 Sort Method: quicksort Memory: 102KB
8 -> Nested Loop (cost=5136.21..2376693.02 rows=4982 width=4) (actual time=234.057..1142.625 rows=1136 loops=1)
9 -> Bitmap Heap Scan on boat b (cost=12.14..74.64 rows=427 width=4) (actual time=239.934..234.488 rows=427 loops=1)
10 Recheck Cond: (color = 'red') bpchar
11 Rows Removed by Index Recheck: 2573
12 Heap Blocks: lossy=25
13 -> Bitmap Index Scan on b_boat (cost=0.00..12.03 rows=3000 width=0) (actual time=0.014..0.014 rows=250 loops=1)
14 Index Cond: (color = 'red') bpchar
15 -> Bitmap Heap Scan on reserves r (cost=5124.07..5565.57 rows=28 width=8) (actual time=0.085..2.123 rows=3 loops=427)
16 Recheck Cond: (bid = b.bid)
17 Rows Removed by Index Recheck: 23677
18 Heap Blocks: lossy=5465
19 -> Bitmap Index Scan on b_reservesbid (cost=0.00..5124.06 rows=35000 width=0) (actual time=0.011..0.011 rows=1280 loops=..)
20 Index Cond: (bid = b.bid)
21 -> Sort (cost=2100061134.96..2100061147.44 rows=4993 width=8) (actual time=3328.951..3329.062 rows=1861 loops=1)
22 Sort Key: s2.sid
23 Sort Method: quicksort Memory: 193KB
24 -> Unique (cost=2100060903.28..2100060828.25 rows=4993 width=8) (actual time=3328.332..3328.698 rows=2064 loops=1)
25 -> Sort (cost=2100060803.28..2100060815.77 rows=4993 width=8) (actual time=3328.330..3328.454 rows=2064 loops=1)
26 Sort Key: s2.sid
27 Sort Method: quicksort Memory: 193KB
28 -> Nested Loop (cost=425148.24..2100060496.57 rows=4993 width=8) (actual time=0.220..3326.738 rows=2064 loops=1)
29 -> Nested Loop (cost=5148.21..2387394.87 rows=4993 width=4) (actual time=0.180..824.069 rows=2064 loops=1)
30 -> Bitmap Heap Scan on boat b2 (cost=12.14..74.64 rows=428 width=4) (actual time=0.082..0.835 rows=428 loops=1)

Total rows: 59 of 59 Query complete 00:00:04.702

Ln 306, Col 1

Activities pgAdmin 4 Jun 9 2:49 AM pgAdmin 4

```

PgAdmin File Object Tools Help
Servers schema3/postgres Local
  + Data Query History
    > p 275 from pg_indexes
    > s 276 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
    > b 277
    > t 278
    > n 279
    > o 280
    > i 281
    > e 282 -- Query 9 (STATISTICS)
    > v 283 set enable_hashagg = off;
    > v 284 set enable_hashjoin = off;
    > v 285 set enable_sort = on;
    > v 286 set enable_seqscan = off;
    > v 287 set enable_bitmapscan = on;
    > v 288
    > v 289
    > v 290 explain analyze select s.sname
    > v 291 from sailors s, reserves r, boat b
    > v 292 where
    > v 293 s.sid = r.sid
    > v 294 and
    > v 295 r.bid = b.bid
    > v 296 and
    > v 297 b.color = 'red'
    > v 298 and
    > v 299 s.sid in ( select s2.sid
    > v 300 from sailors s2, boat b2, reserves r2
    > v 301 where s2.sid = r2.sid
    > v 302 and
    > v 303 r2.bid = b2.bid
    > v 304 and
    > v 305 b2.color = 'green');
    > v 306
    > v 307
    > v 308 -- Query 9 optimization (STATISTICS)
    > v 309 -- Find the names of sailors who have reserved both a red and a green boat.
    > v 310
    > v 311
    > v 312
    > v 313 select *
    > v 314 from pg_indexes
    > v 315 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
    > v 316
  
```

Total rows: 59 of 59 Query complete 00:00:04.702

Ln 306, Col 1

Activities pgAdmin 4 Jun 9 2:50 AM pgAdmin 4

```

PgAdmin File Object Tools Help
Servers schema3/postgres Local
  + Data Query History
    > p 275 from pg_indexes
    > s 276 where tablename = 'sailors' or tablename='reserves'
    > b 277
    > t 278
    > n 279
    > o 280
    > i 281
    > e 282 -- Query 9 (STATISTICS)
    > v 283 set enable_hashagg = off;
    > v 284 set enable_hashjoin = off;
    > v 285 set enable_sort = on;
    > v 286 set enable_seqscan = off;
    > v 287 set enable_bitmapscan = on;
    > v 288
    > v 289
    > v 290 explain analyze select s.sname
    > v 291 from sailors s, reserves r, boat b
    > v 292 where
    > v 293 s.sid = r.sid
    > v 294 and
    > v 295 r.bid = b.bid
    > v 296 and
    > v 297 b.color = 'red'
    > v 298 and
    > v 299 s.sid in ( select s2.sid
    > v 300 from sailors s2, boat b2, reserves r2
    > v 301 where s2.sid = r2.sid
    > v 302 and
    > v 303 r2.bid = b2.bid
    > v 304 and
    > v 305 b2.color = 'green');
    > v 306
    > v 307
    > v 308 -- Query 9 optimization (STATISTICS)
    > v 309 -- Find the names of sailors who have reserved both
    > v 310
    > v 311
    > v 312
    > v 313 select *
    > v 314 from pg_indexes
    > v 315 where tablename = 'sailors' or tablename='reserves'
    > v 316
  
```

Total rows: 1 of 1 Query complete 00:00:04.674

Ln 290, Col 17

Explanation :

- Metrics :

Execution Time : 4679 ms		Total Expected Cost : 2401050935.23	
--------------------------	--	-------------------------------------	--

- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the Worst of all .
- This happened because the Query Optimizer didnt used it from the place due to BRIN Usage here was not suitable so we have used it to simulate seqscan behaviour only we traversed it all and followed all its pointers so it is worst index to use in this case.

5. given query with mixed indices (any mix of your choice).

pgAdmin 4 - Jun 9 3:08 AM ● pgAdmin 4

```

SELECT r.sid
FROM reserves r
WHERE EXISTS
  (SELECT bid
   FROM boat b
   WHERE color = 'red' AND r.bid = b.bid)
  ) AS r2
ON r2.sid = r1.sid ) AS rTotal
WHERE rTotal.sid = s.sid
;

CREATE INDEX b_sailorssID ON sailors USING HASH(sid);
CREATE INDEX b_reservesID ON reserves USING HASH(sid);
CREATE INDEX b_reservesBID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);

CREATE INDEX b_boat2 ON boat USING btree(color);

SELECT *
FROM pg_indexes
WHERE tablename = 'sailors' OR tablename='reserves' OR tablename='boat';

-- Query 9 (STATISTICS)
SET enable_hashagg = off;
SET enable_hashjoin = off;

EXPLAIN ANALYZE SELECT s.sname
FROM sailors s, reserves r, boat b
WHERE
  s.sid = r.sid
  AND
  r.bid = b.bid
  AND
  b.color = 'red';

```

Total rows: 5 of 5 Query complete 00:00:00.207 Ln 274, Col 1

pgAdmin 4 - Jun 9 3:12 AM ● pgAdmin 4

```

CREATE INDEX b_sailorssID ON sailors USING HASH(sid);
CREATE INDEX b_reservesID ON reserves USING HASH(sid);
CREATE INDEX b_reservesBID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);

CREATE INDEX b_boat2 ON boat USING btree(color);

SELECT *
FROM pg_indexes
WHERE tablename = 'sailors' OR tablename='reserves' OR tablename='boat';

-- Query 9 (STATISTICS)
SET enable_hashagg = off;
SET enable_hashjoin = off;

EXPLAIN ANALYZE SELECT s.sname
FROM sailors s, reserves r, boat b
WHERE
  s.sid = r.sid
  AND
  r.bid = b.bid
  AND
  b.color = 'red'
  AND
  s.sid IN (SELECT s2.sid
  FROM sailors s2, boat b2, reserves r2
  WHERE s2.sid = r2.sid
  AND
  r2.bid = b2.bid
  AND
  b2.color = 'green');

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

```

QUERY PLAN

text
1 Nested Loop Semi Join (cost=11.59..3032.13 rows=1309 width=21) (actual time=2,590.8,704 rows=177 loops=1)
2 Join Filter (s.sid = s2.sid)
3 -> Nested Loop (cost=11.59..1987.83 rows=4982 width=29) (actual time=0,055..2,488 rows=1136 loops=1)
4 -> Nested Loop (cost=11.59..1722.72 rows=4982 width=4) (actual time=0,051..1,021 rows=1136 loops=1)
5 -> Bitmap Heap Scan on boat b (cost=11.59..41.93 rows=427 width=4) (actual time=0,043..0,100 rows=427 loops=1)
6 Recheck Cond: (color = 'red')::bpchar
7 Heap Blocks: exact=4
8 -> Bitmap Index Scan on b_boat2 (cost=0,00..11.48 rows=427 width=0) (actual time=0,035..0,035 rows=427 loops=1)
9 Index Cond: (color = 'red')::bpchar
10 -> Index Scan using b_reservesBID on reserves r (cost=0,00..3,66 rows=28 width=8) (actual time=0,001..0,002 rows=3 loops=427)
11 Index Cond: (bid = b.bid)
12 -> Index Scan using b_sailorssID on sailors s (cost=0,00..0,04 rows=1 width=25) (actual time=0,001..0,001 rows=1 loops=1136)
13 Index Cond: (sid = r.sid)
14 -> Nested Loop (cost=0,00..0,20 rows=1 width=8) (actual time=0,005..0,005 rows=0 loops=1136)
15 Join Filter (r2.sid = s2.sid)
16 -> Nested Loop (cost=0,00..0,14 rows=1 width=4) (actual time=0,005..0,005 rows=0 loops=1136)
17 -> Index Scan using b_reservesID on reserves r2 (cost=0,00..0,07 rows=2 width=8) (actual time=0,001..0,002 rows=2 loops=1136)
18 Index Cond: (sid = r2.sid)
19 -> Index Scan using b_boat1 on boat b2 (cost=0,00..0,02 rows=1 width=4) (actual time=0,001..0,001 rows=0 loops=3199)
20 Index Cond: (bid = b2.bid)
21 Filter: (color = 'green')::bpchar
22 Rows Removed by Filter: 1
23 -> Index Scan using b_sailorssID on sailors s2 (cost=0,00..0,04 rows=1 width=4) (actual time=0,001..0,001 rows=1 loops=177)
24 Index Cond: (sid = r2.sid)
25 Planning Time: 0,497 ms
26 Execution Time: 8,749 ms

Total rows: 26 of 26 Query complete 00:00:00.082 Ln 287, Col 1

```

CREATE INDEX b_sailorsSID ON sailors USING hash(sid);
CREATE INDEX b_reservesID ON reserves USING HASH(sid );
CREATE INDEX b_reservesBID ON reserves USING HASH(bid );
CREATE INDEX b_boat1 ON boat USING HASH(bid);
CREATE INDEX b_boat2 ON boat USING btree(color );

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red'
and
s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
and
r2.bid = b2.bid
and
b2.color = 'green');

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

```

Total rows: 1 of 1 Query complete 00:00:00.050 Ln 287, Col 17

Explanation :

- Metrics :

Execution Time : 4679 ms Total Expected Cost : 3032.13
--

- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries.
- Here it showed the improvement due to for every condition of Joining in the Query the Nested Loop Join using index scan using Hash based algorithm which approximatly maded to be O(n) performance.
- The Where clause condition on the color used B-tree which is more better as colors are repeatable and sorted after each other at the leaves of the b-tree with O(Log n) performance on all at one time little bit better than Hash O(1) but calculation of hashfunction and stored in different buckets .

Optimized Query

```

select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
          from

```

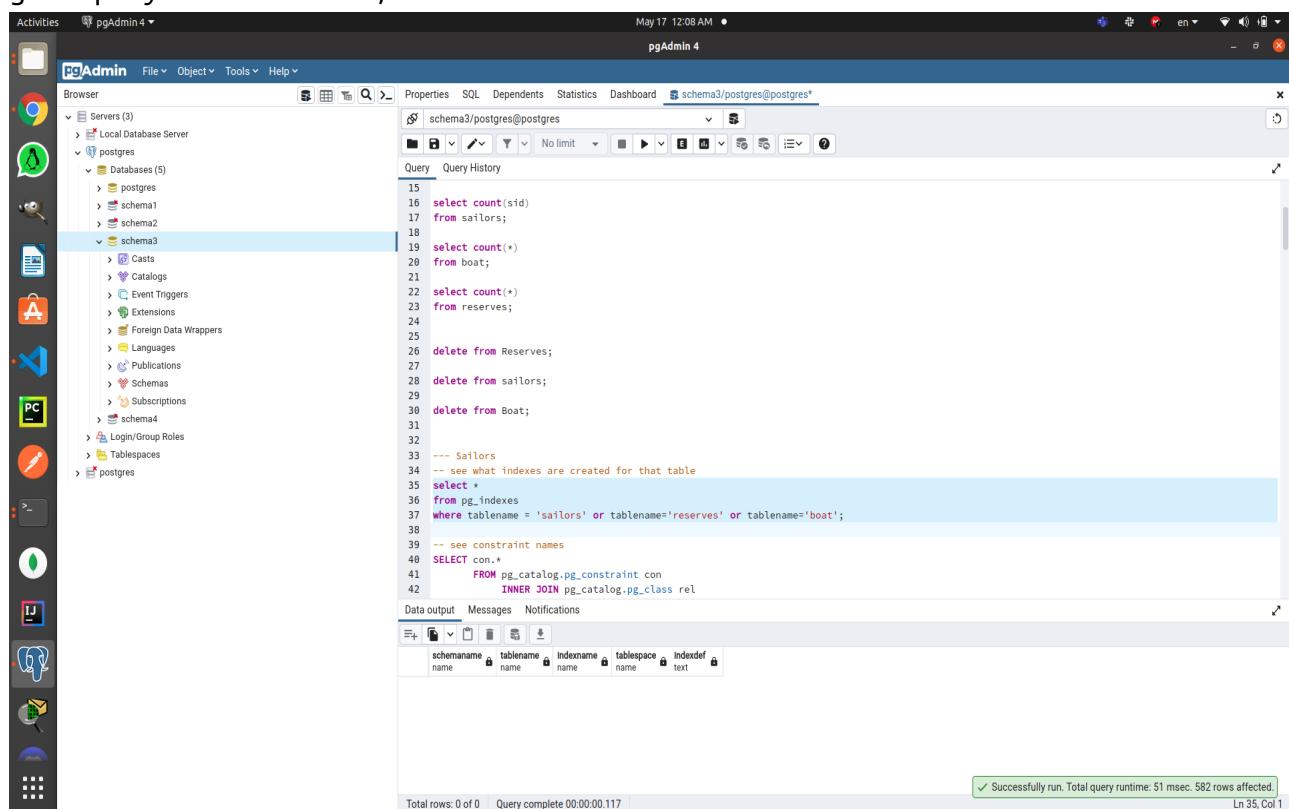
```

(select r.sid
from reserves r
where exists
(select bid
from boat b
where color = 'green' and r.bid =b.bid )
)as r1
inner join
(select r.sid
from reserves r
where exists
(select bid
from boat b
where color = 'red' and r.bid =b.bid )
) as r2
on r2.sid = r1.sid ) as rTotal
where rTotal.sid=s.sid
)

```

Report

1. given query without an index,



The screenshot shows the pgAdmin 4 interface. On the left is a tree view of the database structure under 'Servers (3)'. Under 'postgres', there are 'Databases (5)' including 'postgres', 'schema1', 'schema2', and 'schema3'. 'schema3' is currently selected. The 'Properties' tab is open for 'schema3/postgres@postgres'. In the center is a 'Query' window containing the following SQL code:

```

15 select count(sid)
16   from sailors;
17
18 select count(*)
19   from boat;
20
21 select count(*)
22   from reserves;
23
24
25 delete from Reserves;
26
27
28 delete from sailors;
29
30 delete from Boat;
31
32
33 --- Sailors
34 -- see what indexes are created for that table
35 select *
36   from pg_indexes
37 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
38
39 -- see constraint names
40 SELECT con.*
41   FROM pg_catalog.pg_constraint con
42     INNER JOIN pg_catalog.pg_class rel

```

At the bottom of the query window, it says 'Total rows: 0 of 0 Query complete 00:00:00.117' and 'Successfully run. Total query runtime: 51 msec. 582 rows affected.' Below the query window is a 'Data output' tab.

pgAdmin 4 • Jun 8 11:55 PM

Databases (5) Query Explain Analyze (Shift+F7)

```

300 b2.color = 'green');
301
302
303 -- Query 9 optimization (STATISTICS)
304 -- Find the names of sailors who have reserved both a red and a green boat
305
306 select *
307   from pg_indexes
308  where tablename = 'sailors' or tablename='reserves' or tablename='boat';
309
310 set enable_hashagg = off;
311 set enable_hashjoin = off;
312
313 explain analyze select s.sname
314   from sailors s
315  where exists
316    (
317      select rTotal.sid
318        from (select r1.sid
319              from
320                (select r.sid
321                  from reserves r
322                 where exists
323                   (select bid
324                     from boat b
325                     where color = 'green' and r.bid =b.bid )
326
327               ) as r1
328             inner join
329               (select r.sid
330                 from reserves r
331                 where exists
332                   (select bid
333                     from boat b
334                     where color = 'red' and r.bid =b.bid )
335
336               ) as r2
337             on r2.sid = r1.sid ) as rTotal
338           where rTotal.sid=s.sid
339
340

```

Total rows: 1 of 1 Query complete 00:00:00.081

pgAdmin 4 • Jun 8 11:53 PM

Databases (5) Query Explain Analyze (Shift+F7)

```

300 b2.color = 'green');
301
302
303 -- Query 9 optimization (STATISTICS)
304 -- Find the names of sailors who have reserved both a red and a green boat.
305
306 select *
307   from pg_indexes
308  where tablename = 'sailors' or tablename='reserves' or tablename='boat';
309
310 set enable_hashagg = off;
311 set enable_hashjoin = off;
312
313 explain analyze select s.sname
314   from sailors s
315  where exists
316    (
317      select rTotal.sid
318        from (select r1.sid
319              from
320                (select r.sid
321                  from reserves r
322                 where exists
323                   (select bid
324                     from boat b
325                     where color = 'green' and r.bid =b.bid )
326
327               ) as r1
328             inner join
329               (select r.sid
330                 from reserves r
331                 where exists
332                   (select bid
333                     from boat b
334                     where color = 'red' and r.bid =b.bid )
335
336               ) as r2
337             on r2.sid = r1.sid ) as rTotal
338           where rTotal.sid=s.sid
339
340

```

Total rows: 40 of 40 Query complete 00:00:00.248

QUERY PLAN

- Merge Semi Join (cost=9291.63..9466.30 rows=1324 width=21) (actual time=39.371..40.008 rows=177 loops=1)
 - Merge Cond: (s.sid = r1.sid)
 - Sort (cost=1699.30..1746.80 rows=19000 width=25) (actual time=6.059..6.110 rows=644 loops=1)
 - Sort Key: s.sid
 - Sort Method: quicksort Memory: 2259kB
 - Seq Scan on sailors s (cost=0.00..349.00 rows=19000 width=25) (actual time=0.013..2.765 rows=19000 loops=1)
 - Merge Join (cost=7592.33..7655.45 rows=1324 width=8) (actual time=33.273..33.786 rows=177 loops=1)
 - Merge Cond: (r.sid = r1.sid)
 - Sort (cost=3796.63..3809.11 rows=4993 width=4) (actual time=15.839..16.001 rows=1861 loops=1)
 - Sort Key: r.sid
 - Sort Method: quicksort Memory: 193kB
 - Merge Semi Join (cost=3262.84..3489.91 rows=4993 width=4) (actual time=14.640..15.283 rows=2064 loops=1)
 - Merge Cond: (r.bid = b.bid)
 - Sort (cost=3181.64..3269.14 rows=35000 width=8) (actual time=13.981..14.196 rows=3201 loops=1)
 - Sort Key: r.bid
 - Sort Method: quicksort Memory: 3006kB
 - Seq Scan on reserves r (cost=0.00..540.00 rows=35000 width=8) (actual time=0.011..4.874 rows=35000 loops=1)
 - Sort (cost=81.21..82.28 rows=428 width=4) (actual time=0.486..0.516 rows=428 loops=1)
 - Sort Key: b.bid
 - Filter: (color = 'green'.bpchar)
 - Rows Removed by Filter: 2572
 - Sort (cost=395.70..3808.16 rows=4982 width=4) (actual time=17.369..17.453 rows=1136 loops=1)
 - Sort Key: r_1.sid
 - Sort Method: quicksort Memory: 45kB
 - Seq Scan on boat b (cost=0.00..62.50 rows=428 width=4) (actual time=0.069..0.427 rows=428 loops=1)
 - Merge Semi Join (cost=3262.79..3489.75 rows=4982 width=4) (actual time=16.407..17.021 rows=1136 loops=1)
 - Merge Cond: (r_1.bid = b_1.bid)
 - Sort (cost=3181.64..3269.14 rows=35000 width=8) (actual time=15.586..15.707 rows=1137 loops=1)
 - Sort Key: r_1.bid

Ln 340, Col 1

The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects. In the center is a query editor with the following SQL code:

```

300 b2.color = 'green');
301
302
303 -- Query 9 optimization (STATISTICS)
304 -- Find the names of sailors who have reserved both a red and a green boat.
305
306
307
308 select *
309 from pg_indexes
310 where tablename = 'sailors' or tablename='reserves' or tablename='boat';
311
312 set enable_hashagg = off;
313 set enable_hashjoin = off;
314
315 explain analyze select s.sname
316 from sailors s
317 where exists
318 (
319     select rTotal.sid
320     from (select r1.sid
321           from
322             (select r.sid
323              from reserves r
324              where exists
325                  (select bid
326                   from boat b
327                   where color = 'green' and r.bid =b.bid )
328               )as r1
329             inner join
330               (select r.sid
331                 from reserves r
332                 where exists
333                     (select bid
334                       from boat b
335                       where color = 'red' and r.bid =b.bid )
336                     ) as r2
337             on r2.sid = r1.sid ) as rTotal
338             where rTotal.sid=s.sid
339
340

```

The status bar at the bottom indicates "Total rows: 40 of 40" and "Query complete 00:00:00.248". To the right, a "Data output" tab is active, showing the query plan with 40 numbered steps. The plan details various operations like Merge Semi Join, Sort, and Seq Scan.

Explanation :

- Metrics :

Execution Time : 40.659 ms	Total Expected Cost : 9466.30
----------------------------	-------------------------------

- Same flags is set to all here too.

- Reason :

- This Query Improved in the Execution time and Expected Cost than the Original Query from 11368.59 to 9466.30.
- Because the loops ends faster and exits due to I used the Exist Operator instead of the In Operator so it helped in the intermediate results.
- I have used the table Sailors once instead of twiceas used in original Query .
- 6 less steps of the Query Plan where made in the Optimized one which helped in decreasing the cost.

2. given query with B+ trees indices only,

Activities pgAdmin 4 Jun 9 12:37 AM ● pgAdmin 4

Servers schema3/postgres Local Data Query History Execute/Refresh

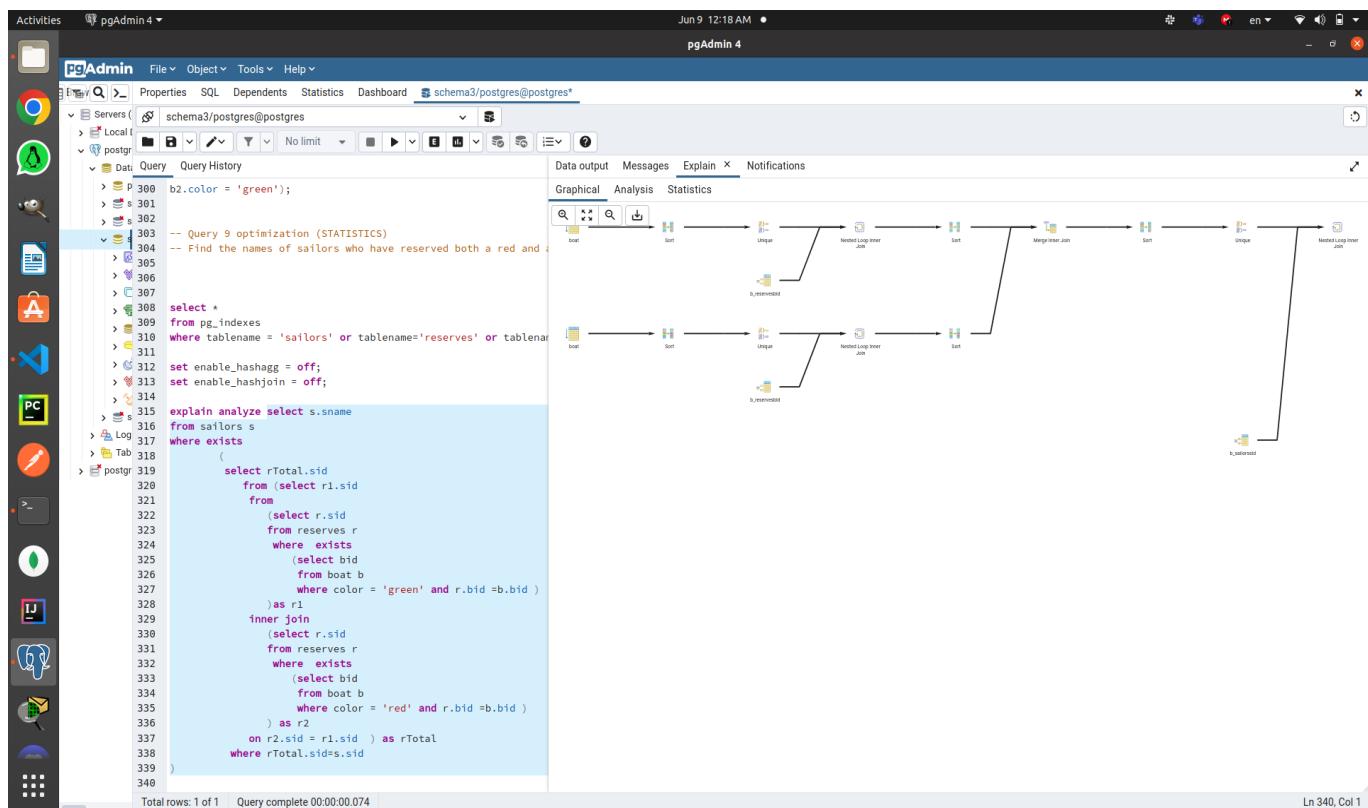
```

> P 300 b2.color = 'green';
> S 301
> S 302
> S 303 -- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

> S 304
> S 305
> S 306
> S 307
> S 308 select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';
> S 309
> S 310
> S 311 set enable_hashagg = off;
> S 312 set enable_hashjoin = off;
> S 313
> S 314
> S 315 explain analyze select s.sname
from sailors s
where exists
(
> S 316
> S 317
> S 318
> S 319
> S 320
> S 321
> S 322
> S 323
> S 324
> S 325
> S 326
> S 327
> S 328
> S 329
> S 330
> S 331
> S 332
> S 333
> S 334
> S 335
> S 336
> S 337
> S 338
> S 339
> S 340
)
select rTotal.sid
from (select r1.sid
from
(select r.sid
from reserves r
where exists
(
select bid
from boat b
where color = 'green' and r.bid = b.bid
)
as r1
inner join
(select r.sid
from reserves r
where exists
(
select bid
from boat b
where color = 'red' and r.bid = b.bid
)
) as r2
on r2.sid = r1.sid ) as rTotal
where rTotal.sid=s.sid
)

```

Total rows: 4 of 4 Query complete 00:00:00.224 Ln 308, Col 1



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure with 'Servers (3)' and 'PostgreSQL' selected. The main area contains a query editor with the following SQL script:

```
s.sid in ( select s2.sid
 286   from sailors s2, boat b2, reserves r2
 287   where s2.sid = r2.sid
 288   and
 289   r2.bid = b2.bid
 290   and
 291   b2.color = 'green');
 292 Event Ti_292
 293 Extensi_293
 294 Foreign_294 -- Query 9 optimization (STATISTICS)
 295 Public_297 -- Find the names of sailors who have reserved both a red and a green boat.
 296 Langu_296
 297 set enable_hashjoin= off;
 298 set enable_hashagg= off;
 299 Subscr_299
 300
 301 explain analyze select s.sname
 302   from sailors s
 303   where exists
 304     (
 305       select rTotal.sid
 306         from (select r1.sid
 307           from
 308             (select r.sid
 309               from reserves r
 310               where exists
 311                 (select bid
 312                   from boat b
 313                   where color = 'green' and r.bid = b.bid )
 314                 ) as r1
 315                 inner join
 316                   (select r.sid
 317                     from reserves r
 318                     where exists
 319                       (select bid
 320                         from boat b
 321                         where color = 'red' and r.bid = b.bid )
 322                   ) as r2
 323                   on r2.sid = r1.sid ) as rTotal
 324   where rTotal.sid=s.sid
 325
 326 Total rows: 37 of 37  Query complete 00:00:00.099
```

The right panel shows the 'Data output' tab with the 'QUERY PLAN' section expanded, displaying the execution plan for the query. The plan includes various stages such as Sort, Nested Loop, Unique, and Seq Scan, along with their respective costs, row counts, and execution times.

Explanation :

- Metrics :

| Execution Time : 4.890 ms | Total Expected Cost : 4695.10 |

- The B-tree helped in the performance it decreased the Execution Time and Expected Cost .
- The Query Planner used the B-tree index because B-Tree is O(Log n) performance with Exact Values
- Here it showed the improvement due to for every condition of Joining in the Query the Merge Join used index scan using ZigZag algorithm .

3. given query with hash indices only,

pgAdmin 4 - Jun 9 12:48 AM

```

inner join
  (select r.sid
   from reserves r
   where exists
     (select bid
      from boat b
      where color = 'red' and r.bid = b.bid)
    ) as r2
  on r2.sid = r1.sid ) as rTotal
where rTotal.sid=s.sid

CREATE INDEX b_sailorssid ON sailors USING HASH(sid);
CREATE INDEX b_reservesSID ON reserves USING HASH(sid);
CREATE INDEX b_reservesBID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);
CREATE INDEX b_boat2 ON boat USING HASH(color);

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid;

```

Total rows: 5 of 5 Query complete 00:00:00.050

pgAdmin 4 - Jun 9 1:14 AM

```

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

set enable_hashagg = off;
set enable_hashjoin = off;

explain analyze select sname
from sailors s
where exists
(
  select rTotal.sid
  from (select r1.sid
        from
          (select r.sid
           from reserves r
           where exists
             (select bid
              from boat b
              where color = 'green' and r.bid = b.bid)
            ) as r1
        inner join
          (select r.sid
           from reserves r
           where exists
             (select bid
              from boat b
              where color = 'red' and r.bid = b.bid)
            ) as r2
        on r2.sid = r1.sid ) as rTotal
  where rTotal.sid=s.sid
)


```

Total rows: 23 of 23 Query complete 00:00:00.176

```

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat
select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boats'
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
        from reserves r
        where exists
        (select bid
        from boat b
        where color = 'green' and r.bid = b.bid )
        ) as r1
    inner join
        (select r.sid
        from reserves r
        where exists
        (select bid
        from boat b
        where color = 'red' and r.bid = b.bid )
        ) as r2
    on r2.sid = r1.sid ) as rTotal
    where rTotal.sid=s.sid
)

```

Total rows: 1 of 1 Query complete 00:00:00.218

Successfully run. Total query runtime: 218 msec. 1 rows affected.

Ln 344, Col 1

- Metrics :

Execution Time : 12.646 ms	Total Expected Cost : 2337.61
----- -----	

- The Hash helped in the performance it decreased the Execution Time (but in the execution time processor was overwhelmed + Linux performance) and Expected Cost .
- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries.
- Here it showed the improvement due to for every condition of Joining in the Query the Nested Loop Join using index scan using Hash based algorithm which approximatlly maded to be O(n) performance.
- The Where clause condition on the color used Hash too.

4. given query with BRIN indices only,

pgAdmin 4 - Jun 9 2:29 AM

```

CREATE INDEX b_sailorsSID ON sailors USING BRIN(sid );
CREATE INDEX b_reservesSID ON reserves USING BRIN(sid );
CREATE INDEX b_reservesBID ON reserves USING BRIN(bid );
CREATE INDEX b_boat ON boat USING BRIN(bid,color );

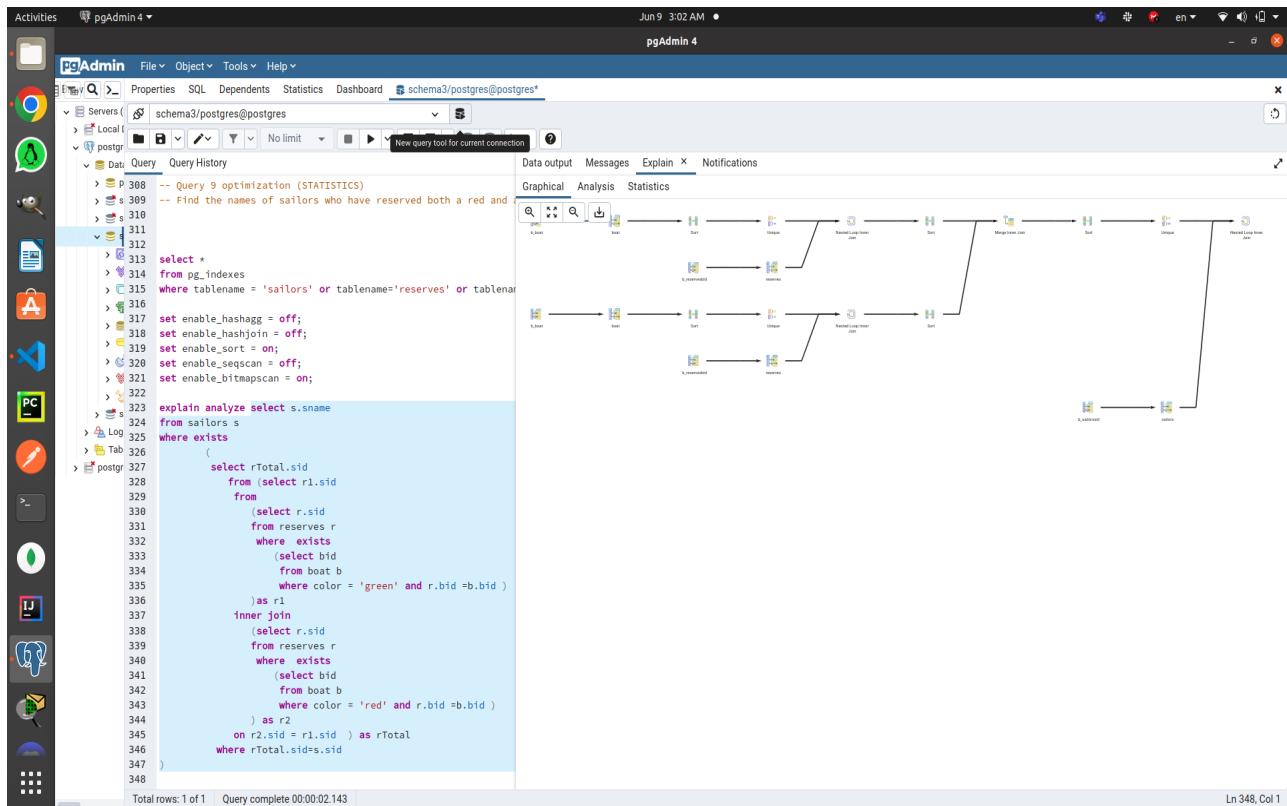
select *
from pg_indexes
where tablename = 'sailors' or tablename='reserves' or tablename='boat';

-- Query 9 (STATISTICS)
set enable_hashagg = off;
set enable_hashjoin = off;
explain analyze select s.sname
from sailors s, reserves r, boat b
where
s.sid = r.sid
and
r.bid = b.bid
and
b.color = 'red'
and
s.sid in ( select s2.sid
from sailors s2, boat b2, reserves r2
where s2.sid = r2.sid
and
r2.bid = b2.bid
and
b2.color = 'green');

```

Total rows: 4 of 4 Query complete 00:00:00.088

Ln 273, Col 1



Activities pgAdmin 4 Jun 9 3:01 AM pgAdmin 4

```

P 308 -- Query 9 optimization (STATISTICS)
s 309 -- Find the names of sailors who have reserved both
e 310
i 311
o 312
n 313
c 314
t 315
r 316
d 317
u 318
s 319
l 320
f 321
g 322
h 323
j 324
k 325
m 326
p 327
q 328
w 329
x 330
y 331
z 332
A 333
B 334
C 335
D 336
E 337
F 338
G 339
H 340
I 341
J 342
K 343
L 344
M 345
N 346
O 347
P 348

explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
        from reserves r
        where exists
            (select bid
            from boat b
            where color = 'green' and r.bid = r1.sid
            ) as r1
        inner join
        (select r.sid
        from reserves r
        where exists
            (select bid
            from boat b
            where color = 'red' and r.bid = r1.sid
            ) as r2
        on r2.sid = r1.sid ) as rTotal
        where rTotal.sid=s.sid
    )
)
)
Total rows: 59 of 59 Query complete 00:00:02.186

```

Activities pgAdmin 4 Jun 9 3:01 AM pgAdmin 4

```

P 308 -- Query 9 optimization (STATISTICS)
s 309 -- Find the names of sailors who have reserved both
e 310
i 311
o 312
n 313
c 314
t 315
r 316
d 317
u 318
s 319
l 320
f 321
g 322
h 323
j 324
k 325
m 326
p 327
q 328
w 329
x 330
y 331
z 332
A 333
B 334
C 335
D 336
E 337
F 338
G 339
H 340
I 341
J 342
K 343
L 344
M 345
N 346
O 347
P 348

explain analyze select s.sname
from sailors s
where exists
(
    select rTotal.sid
    from (select r1.sid
    from
        (select r.sid
        from reserves r
        where exists
            (select bid
            from boat b
            where color = 'green' and r.bid = r1.sid
            ) as r1
        inner join
        (select r.sid
        from reserves r
        where exists
            (select bid
            from boat b
            where color = 'red' and r.bid = r1.sid
            ) as r2
        on r2.sid = r1.sid ) as rTotal
        where rTotal.sid=s.sid
    )
)
)
Total rows: 59 of 59 Query complete 00:00:02.186

```

Explanation :

- Metrics :

Execution Time : 4679 ms Total Expected Cost : 303415313.52

- Here the BRIN was not used in the original Query Plan settings (Hashjoin and HashAgg are off) so I've made seqscan=off too.
- The Execution Time and Expected Cost became the Worst of all .
- This happened because the Query Optimizer didnt used it from the place due to BRIN Usage here was not suitable so we have used it to .

5. given query with mixed indices (any mix of your choice)

pgAdmin 4 - Jun 9 3:08 AM • pgAdmin 4

```

SELECT r.sid
FROM reserves r
WHERE EXISTS
    (SELECT bid
     FROM boat b
     WHERE color = 'red' AND r.bid = b.bid)
    ) AS r2
ON r2.sid = r1.sid ) AS rTotal
WHERE rTotal.sid = s.sid
;

CREATE INDEX b_sailorssid ON sailors USING HASH(sid);
CREATE INDEX b_reservesSID ON reserves USING HASH(sid);
CREATE INDEX b_reservesBID ON reserves USING HASH(bid);
CREATE INDEX b_boat1 ON boat USING HASH(bid);

CREATE INDEX b_boat2 ON boat USING btree(color);

SELECT *
FROM pg_indexes
WHERE tablename = 'sailors' OR tablename='reserves' OR tablename='boat';
;

-- Query 9 (STATISTICS)
SET enable_hashagg = off;
SET enable_hashjoin = off;

EXPLAIN ANALYZE SELECT s.sname
FROM sailors s, reserves r, boat b
WHERE
    s.sid = r.sid
    AND
    r.bid = b.bid
    AND
    b.color = 'red'
;

```

Total rows: 5 of 5 Query complete 00:00:00.207 Ln 274, Col 1

pgAdmin 4 - Jun 9 3:27 AM • pgAdmin 4

```

SELECT b2.color = 'green';
;

-- Find the names of sailors who have reserved both a red and a green boat.
;

-- Query 9 optimization (STATISTICS)
-- Find the names of sailors who have reserved both a red and a green boat.

SELECT *
FROM pg_indexes
WHERE tablename = 'sailors' OR tablename='reserves' OR tablename='boat';

SET enable_hashagg = off;
SET enable_hashjoin = off;

EXPLAIN ANALYZE SELECT s.sname
FROM sailors s
WHERE EXISTS
(
    SELECT rTotal.sid
    FROM (SELECT r1.sid
          FROM
              (SELECT r.sid
               FROM reserves r
               WHERE EXISTS
                   (SELECT bid
                    FROM boat b
                    WHERE color = 'green' AND r.bid = b.bid)
                   ) AS r1
          INNER JOIN
              (SELECT r.sid
               FROM reserves r
               WHERE EXISTS
                   (SELECT bid
                    FROM boat b
                    WHERE color = 'red' AND r.bid = b.bid)
                   ) AS r2
          ON r2.sid = r1.sid ) AS rTotal
    WHERE rTotal.sid = s.sid
);

```

Total rows: 1 of 1 Query complete 00:00:00.165 Ln 342, Col 1



Explanation :

- Metrics :

Execution Time : 4679 ms	Total Expected Cost : 2337.61
----- -----	

- The Query Planner used the Hash index because Hash is O(1) performance with Exact Values and considered the best for exact values queries.
- Here it showed the improvement due to for every condition of Joining in the Query the Nested Loop Join using index scan using Hash based algorithm which approximatly maded to be O(n) performance.
- The Where clause condition on the color used B-tree which is more better as colors are repeatable and sorted after each other at the leaves of the b-tree with O(Log n) performance on all at one time little bit better than Hash O(1) but calculation of hashfunction and stored in different buckets .