PG_DWH TASK 6

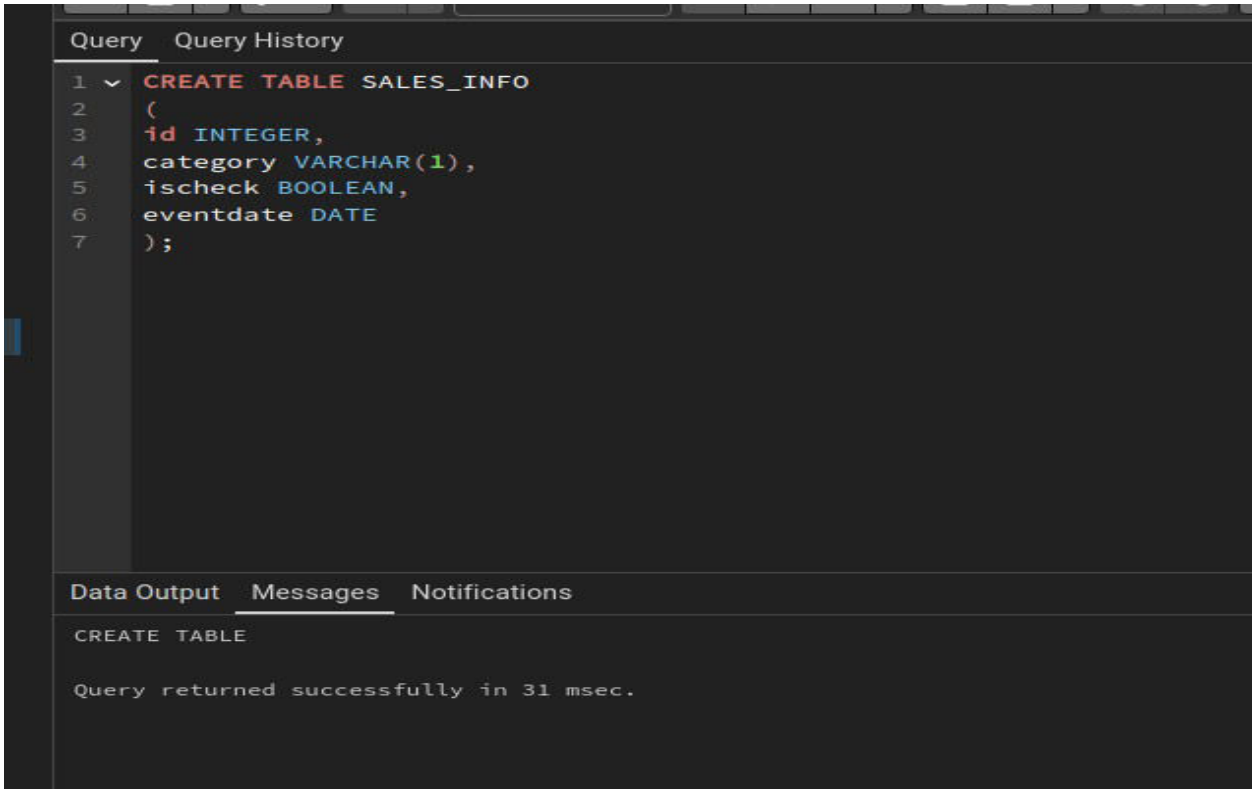
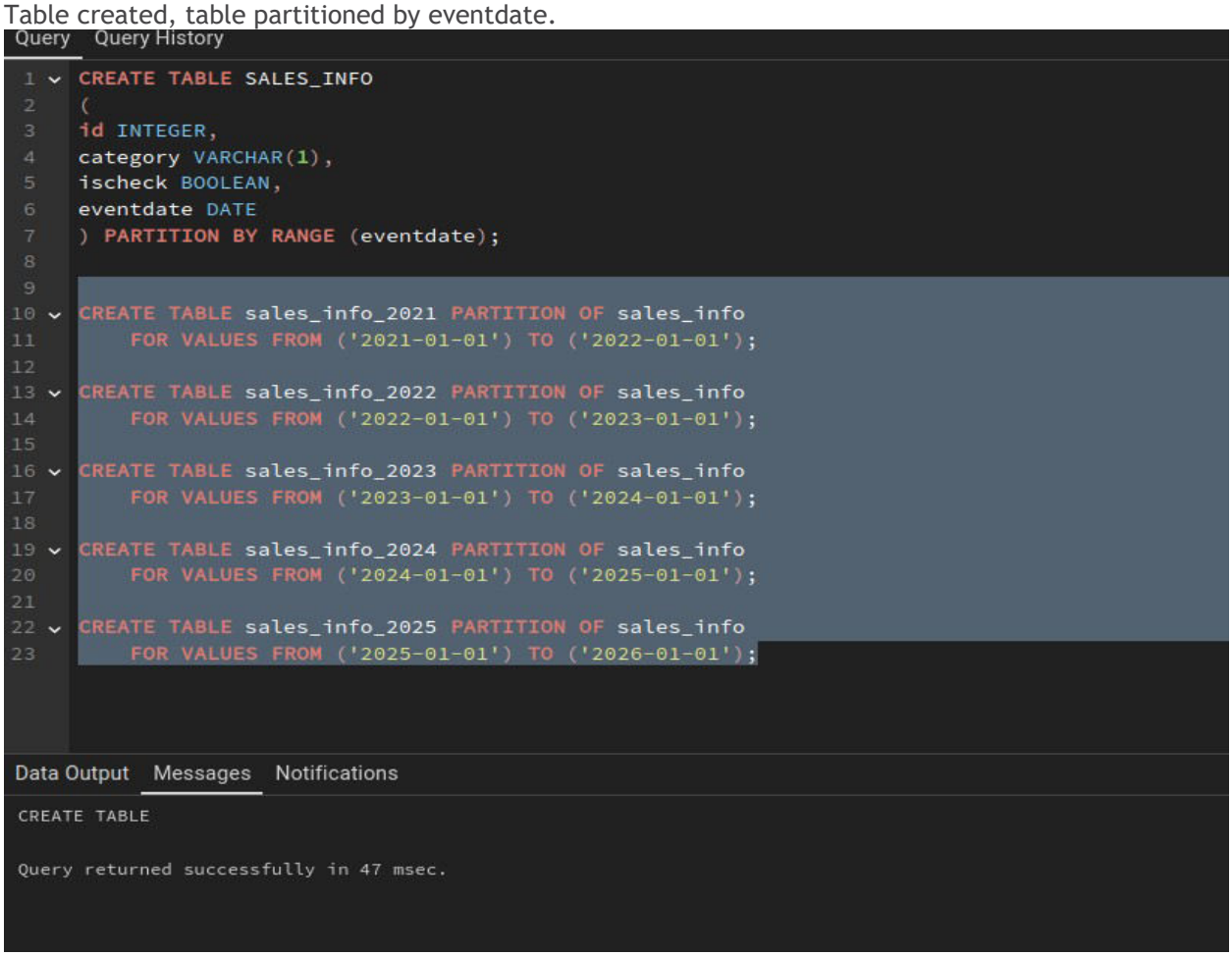


Table created, table partitioned by eventdate.



```
CREATE OR REPLACE FUNCTION partition_sales_info() RETURNS trigger
AS $$
BEGIN
    IF (new.eventdate >= '2022-01-01'::DATE AND new.eventdate < '2023-01-01'::DATE) THEN
        INSERT INTO sales_info_2022 VALUES (new.*);
    ELSIF (new.eventdate >= '2021-01-01'::DATE AND new.eventdate < '2022-01-01'::DATE) THEN
        INSERT INTO sales_info_2021 VALUES (new.*);
    ELSIF (new.eventdate >= '2023-01-01'::DATE AND new.eventdate < '2024-01-01'::DATE) THEN
        INSERT INTO sales_info_2023 VALUES (new.*);
    ELSIF (new.eventdate >= '2024-01-01'::DATE AND new.eventdate < '2025-01-01'::DATE) THEN
        INSERT INTO sales_info_2024 VALUES (new.*);
    ELSIF (new.eventdate >= '2025-01-01'::DATE AND new.eventdate < '2026-01-01'::DATE) THEN
        INSERT INTO sales_info_2025 VALUES (new.*);
    ELSE
        RAISE EXCEPTION 'Out of range';
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 38 msec.
```

Function created for partitioning.

Trigger to use the function partition_sales_info on sales_info table , it will be called before inserting row in the table.

```
CREATE TRIGGER partition_sales_info_trigger
    BEFORE INSERT ON sales_info

    FOR EACH ROW
    WHEN (pg_trigger_depth() < 1)
    EXECUTE PROCEDURE partition_sales_info();
```

I got an error before used WHEN condition , it was inserting the same row over and over , idk why but I found solution by using WHEN condition here .
Then inserted all rows.

```
SET max_stack_depth = 7680

INSERT INTO sales_info(id, category, ischeck, eventdate)
SELECT id,
       (ARRAY['A','B','C','D','E','F','J','H','I','J','K'])[((RANDOM())*10)::INTEGER] AS category,
       ((1*(RANDOM())::INTEGER)<1) AS ischeck,
       (NOW() - '10 day'::INTERVAL * (RANDOM()::int * 100))::DATE AS eventdate
FROM generate_series(1, 10000000) AS id;
```

Data Output | Messages | Notifications

```
INSERT 0 10000000

Query returned successfully in 8 secs 270 msec.
```

```
66  FROM generate_series(1, 10000000) AS id;
67
68
69  UPDATE sales_info
70  SET eventdate = '2024-07-15'
71  WHERE id <= 1000;
72
73
```

Data Output | Messages | Notifications

```
UPDATE 1000

Query returned successfully in 377 msec.
```

Update some rows.

```
85
86
87
88
89  EXPLAIN ANALYZE SELECT * FROM sales_info_simple;
90  EXPLAIN ANALYZE SELECT * FROM sales_info;
91
92  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on sales_info_simple (cost=0.00..154056.75 rows=10000175 width=11) (actual time=0.014..426.037 rows=10000000 loop...
2	Planning Time: 0.067 ms
3	Execution Time: 614.164 ms

Query plan for sales_info_sample and for sales_info.

```
89  EXPLAIN ANALYZE SELECT * FROM sales_info_simple;
90  EXPLAIN ANALYZE SELECT * FROM sales_info;
91
92  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..204189.10 rows=10006540 width=11) (actual time=0.039..791.876 rows=10000000 loops=1)
2	-> Seq Scan on sales_info_2021 sales_info_1 (cost=0.00..77023.01 rows=4999701 width=11) (actual time=0.038..227.915 rows=4999206 loop...
3	-> Seq Scan on sales_info_2022 sales_info_2 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.005..0.006 rows=0 loops=1)
4	-> Seq Scan on sales_info_2023 sales_info_3 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.001..0.001 rows=0 loops=1)
5	-> Seq Scan on sales_info_2024 sales_info_4 (cost=0.00..77049.09 rows=5001409 width=11) (actual time=0.023..216.664 rows=5000794 loop...
6	-> Seq Scan on sales_info_2025 sales_info_5 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.008..0.008 rows=0 loops=1)
7	Planning Time: 0.076 ms
8	Execution Time: 973.548 ms

As we can see it uses Sequential Scan for whole table which is not partitioned, in second table it scanes separately in each partition.

```
91
92  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
93  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
94
95  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate = '2023-07-15';
96  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate = '2023-07-15';
97
98  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
99  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;
100
101 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
102 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
103
104
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Gather (cost=1000.00..117556.19 rows=1 width=11) (actual time=200.648..204.951 rows=0 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on sales_info_simple (cost=0.00..116556.09 rows=1 width=11) (actual time=187.323..187.324 rows=0 loop...
5	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
6	Rows Removed by Filter: 3333333
7	Planning Time: 0.056 ms
8	JIT:
9	Functions: 6
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 0.897 ms, Inlining 0.000 ms, Optimization 0.549 ms, Emission 8.661 ms, Total 10.108 ms
12	Execution Time: 205.234 ms

Plan with date fillter in not partitioned table.

```
91
92  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
93  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
94
95  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate = '2023-07-15';
96  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate = '2023-07-15';
97
98  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
99  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;
100
101 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31
102 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
103
104
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on sales_info_2023 sales_info (cost=0.00..37.15 rows=9 width=17) (actual time=0.003..0.003 rows=0 loops...
2	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
3	Planning Time: 0.162 ms
4	Execution Time: 0.012 ms

For partitioned one we can see that its only scan on the one partition where the date is included , in not partitioned table it’s need to check whole table.

```
93  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
94
95  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate = '2023-07-15';
96  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate = '2023-07-15';
97
98  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
99  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;
100
101 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-3
102 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
103
104
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Gather (cost=1000.00..107139.34 rows=1 width=11) (actual time=177.447..181.993 rows=0 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on sales_info_simple (cost=0.00..106139.24 rows=1 width=11) (actual time=165.037..165.038 rows=0 loop...
5	Filter: (eventdate = '2023-07-15'::date)
6	Rows Removed by Filter: 3333333
7	Planning Time: 0.053 ms
8	JIT:
9	Functions: 6
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 0.804 ms, Inlining 0.000 ms, Optimization 0.570 ms, Emission 8.443 ms, Total 9.817 ms
12	Execution Time: 182.306 ms

```
93  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
94
95  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate = '2023-07-15';
96  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate = '2023-07-15';
97
98  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
99  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;
100
101 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-3
102 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
103
104
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on sales_info_2023 sales_info (cost=0.00..32.62 rows=9 width=17) (actual time=0.004..0.005 rows=0 loops...
2	Filter: (eventdate = '2023-07-15'::date)
3	Planning Time: 0.073 ms
4	Execution Time: 0.016 ms

Same thing here , we can see that partitioned table is way efficient.

```
93  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
94
95  EXPLAIN ANALYZE SELECT * FROM sales_info_simple WHERE eventdate = '2023-07-15';
96  EXPLAIN ANALYZE SELECT * FROM sales_info WHERE eventdate = '2023-07-15';
97
98  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
99  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;
100
101 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-3
102 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
103
104
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Finalize Aggregate (cost=107139.46..107139.47 rows=1 width=8) (actual time=263.200..267.687 rows=1 loops=1)
2	-> Gather (cost=107139.25..107139.46 rows=2 width=8) (actual time=263.104..267.677 rows=3 loops=1)
3	Workers Planned: 2
4	Workers Launched: 2
5	-> Partial Aggregate (cost=106139.25..106139.26 rows=1 width=8) (actual time=251.017..251.017 rows=1 loops=3)
6	-> Parallel Seq Scan on sales_info_simple (cost=0.00..95722.40 rows=4166740 width=0) (actual time=0.016..156.895 rows=3333333 loop...
7	Planning Time: 0.322 ms
8	JIT:
9	Functions: 8
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 0.423 ms, Inlining 0.000 ms, Optimization 0.277 ms, Emission 5.559 ms, Total 6.259 ms
12	Execution Time: 267.916 ms

S

```
98  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
99  EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;
100
101 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-
102 EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
103
104
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Finalize Aggregate (cost=128064.88..128064.89 rows=1 width=8) (actual time=449.458..452.396 rows=1 loops=1)
2	-> Gather (cost=128064.67..128064.88 rows=2 width=8) (actual time=449.326..452.377 rows=3 loops=1)
3	Workers Planned: 2
4	Workers Launched: 2
5	-> Partial Aggregate (cost=127064.67..127064.68 rows=1 width=8) (actual time=436.684..436.687 rows=1 loops=3)
6	-> Parallel Append (cost=0.00..116641.19 rows=4169391 width=0) (actual time=3.047..336.050 rows=3333333 loops=3)
7	-> Parallel Seq Scan on sales_info_2024 sales_info_4 (cost=0.00..47874.20 rows=2083920 width=0) (actual time=0.739..82.354 rows=1666931 loop...
8	-> Parallel Seq Scan on sales_info_2021 sales_info_1 (cost=0.00..47858.09 rows=2083209 width=0) (actual time=3.502..162.601 rows=2499603 loo...
9	-> Parallel Seq Scan on sales_info_2022 sales_info_2 (cost=0.00..20.65 rows=1065 width=0) (actual time=0.000..0.000 rows=0 loops=1)
10	-> Parallel Seq Scan on sales_info_2023 sales_info_3 (cost=0.00..20.65 rows=1065 width=0) (actual time=0.000..0.000 rows=0 loops=1)
11	-> Parallel Seq Scan on sales_info_2025 sales_info_5 (cost=0.00..20.65 rows=1065 width=0) (actual time=0.002..0.002 rows=0 loops=1)
12	Planning Time: 0.086 ms
13	JIT:
14	Functions: 23
15	Options: Inlining false, Optimization false, Expressions true, Deforming true
16	Timing: Generation 0.877 ms, Inlining 0.000 ms, Optimization 0.466 ms, Emission 8.281 ms, Total 9.623 ms
17	Execution Time: 452.854 ms

When we select count in partitioned table , it’s going to be slower than not partitioned one because in not partitioned table it goes only through one table , but in partitioned one each partition is separate table .

```
EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;

EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-
EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=117556.20..117556.21 rows=1 width=8) (actual time=208.813..213.577 rows=1 loops=1)
2	-> Gather (cost=1000.00..117556.19 rows=1 width=0) (actual time=208.777..213.540 rows=0 loops=1)
3	Workers Planned: 2
4	Workers Launched: 2
5	-> Parallel Seq Scan on sales_info_simple (cost=0.00..116556.09 rows=1 width=0) (actual time=196.512..196.512 rows=0 loop…
6	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
7	Rows Removed by Filter: 3333333
8	Planning Time: 0.061 ms
9	JIT:
10	Functions: 11
11	Options: Inlining false, Optimization false, Expressions true, Deforming true
12	Timing: Generation 1.027 ms, Inlining 0.000 ms, Optimization 0.593 ms, Emission 9.895 ms, Total 11.515 ms
13	Execution Time: 214.039 ms

S

```
EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple;
EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info;

EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info_simple WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
EXPLAIN ANALYZE SELECT COUNT(*) FROM sales_info WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=37.17..37.18 rows=1 width=8) (actual time=0.007..0.007 rows=1 loops=1)
2	-> Seq Scan on sales_info_2023 sales_info (cost=0.00..37.15 rows=9 width=0) (actual time=0.004..0.004 rows=0 loop…
3	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
4	Planning Time: 0.104 ms
5	Execution Time: 0.025 ms

Here we can see that if we have dates , even with count , it will firstly filter by date and choose table to go though , so as we see it’s perfectly optimizes our query.

```
104
105
106 DROP TABLE sales_info_2021;
107
108 CREATE TABLE sales_info_2026 PARTITION OF sales_info
109     FOR VALUES FROM ('2026-01-01') TO ('2027-01-01');
110
111
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 79 msec.
```

New partition created.

Task_2
Table created ,declarative partitions created for each year and categories.

postgres/postgres@DWH

No limit

Query | Query History

```
55 FOR VALUES IN ('DEFAULT');
56
57 -- Sub-partitions for 2023
58 CREATE TABLE SALES_INFO_DP_2023_CAT_A_B PARTITION OF SALES_INFO_DP_2023
59 FOR VALUES IN ('A', 'B');
60
61 CREATE TABLE SALES_INFO_DP_2023_CAT_C_D_E_F PARTITION OF SALES_INFO_DP_2023
62 FOR VALUES IN ('C', 'D', 'E', 'F',null);
63
64 CREATE TABLE SALES_INFO_DP_2023_DEFAULT PARTITION OF SALES_INFO_DP_2023
65 FOR VALUES IN ('DEFAULT');
66
67 -- Sub-partitions for 2024
68 CREATE TABLE SALES_INFO_DP_2024_CAT_A_B PARTITION OF SALES_INFO_DP_2024
69 FOR VALUES IN ('A', 'B');
70
71 CREATE TABLE SALES_INFO_DP_2024_CAT_C_D_E_F PARTITION OF SALES_INFO_DP_2024
72 FOR VALUES IN ('C', 'D', 'E', 'F',null);
73
74 CREATE TABLE SALES_INFO_DP_2024_DEFAULT PARTITION OF SALES_INFO_DP_2024
75 FOR VALUES IN ('DEFAULT');
76
77
78
79 INSERT INTO SALES_INFO_DP(id,category, ischeck, EventDate)
80 SELECT id
81 ,('{"A","B","C","D","E","F"}'::text[])[((
82 RANDOM())*10)::INTEGER] category
83  ,((1*(RANDOM())::INTEGER)<1) ischeck
84  ,(NOW() - '10 day'::INTERVAL * (RANDOM()::int * 100))::
85 DATE EventDate
86 FROM generate_series(1,10000000) id;
```

Data Output | Messages | Notifications

```
INSERT 0 10000000

Query returned successfully in 8 secs 788 msec.
```

```
86  FROM generate_series(1,10000000) id;
87
88  UPDATE SALES_INFO_DP
89  SET category = 'B'
90  WHERE category = 'A' AND eventdate >= '2021-01-01' AND eventdate < '2022-01-01';
91
92
93
```

Data Output | Messages | Notifications

```
UPDATE 499863
```

Updated some records.

```
96  -- 5.1 Select all rows
97  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP;
98
99  -- 5.2 Select with a range of dates
100 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
101
102 -- 5.3 Select an exact date
103 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate = '2023-07-01';
104
105 -- 5.4 Select an exact category
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..206877.40 rows=10002560 width=11) (actual time=0.006..773.888 rows=10000000 loops=1)
2	-> Seq Scan on sales_info_dp_2021_cat_a_b sales_info_dp_1 (cost=0.00..18090.79 rows=998879 width=11) (actual time=0.006..43.141 rows=998879 loops=1)
3	-> Seq Scan on sales_info_dp_2021_cat_c_d_e_f sales_info_dp_2 (cost=0.00..61640.88 rows=4001188 width=11) (actual time=0.173..172.614 rows=4001188 loops...
4	-> Seq Scan on sales_info_dp_2021_default sales_info_dp_3 (cost=0.00..13.20 rows=320 width=227) (actual time=0.007..0.007 rows=0 loops=1)
5	-> Seq Scan on sales_info_dp_2022_cat_a_b sales_info_dp_4 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
6	-> Seq Scan on sales_info_dp_2022_cat_c_d_e_f sales_info_dp_5 (cost=0.00..13.20 rows=320 width=227) (actual time=0.002..0.002 rows=0 loops=1)
7	-> Seq Scan on sales_info_dp_2022_default sales_info_dp_6 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
8	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_7 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
9	-> Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_8 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
10	-> Seq Scan on sales_info_dp_2023_default sales_info_dp_9 (cost=0.00..13.20 rows=320 width=227) (actual time=0.002..0.002 rows=0 loops=1)
11	-> Seq Scan on sales_info_dp_2024_cat_a_b sales_info_dp_10 (cost=0.00..15403.37 rows=999837 width=11) (actual time=0.017..41.955 rows=999837 loops=1)
12	-> Seq Scan on sales_info_dp_2024_cat_c_d_e_f sales_info_dp_11 (cost=0.00..61623.96 rows=4000096 width=11) (actual time=0.025..170.367 rows=4000096 loop...
13	-> Seq Scan on sales_info_dp_2024_default sales_info_dp_12 (cost=0.00..13.20 rows=320 width=227) (actual time=0.007..0.007 rows=0 loops=1)
14	Planning Time: 0.099 ms
15	Execution Time: 954.837 ms

Query plan for our table , each partitions considered separate table.

```
96  -- 5.1 Select all rows
97  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP;
98
99  -- 5.2 Select with a range of dates
100 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
101
102 -- 5.3 Select an exact date
103 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate = '2023-07-01';
104
105 -- 5.4 Select an exact category
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..44.43 rows=6 width=227) (actual time=0.006..0.007 rows=0 loops=1)
2	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_1 (cost=0.00..14.80 rows=2 width=227) (actual time=0.004..0.004 rows=0 loops=1)
3	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
4	-> Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_2 (cost=0.00..14.80 rows=2 width=227) (actual time=0.001..0.001 rows=0 loop...
5	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
6	-> Seq Scan on sales_info_dp_2023_default sales_info_dp_3 (cost=0.00..14.80 rows=2 width=227) (actual time=0.000..0.001 rows=0 loops=1)
7	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
8	Planning Time: 0.132 ms
9	Execution Time: 0.022 ms

```
96  -- 5.1 Select all rows
97  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP;
98
99  -- 5.2 Select with a range of dates
100 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
101
102 -- 5.3 Select an exact date
103 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate = '2023-07-01';
104
105 -- 5.4 Select an exact category
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..42.03 rows=6 width=227) (actual time=0.007..0.008 rows=0 loops=1)
2	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_1 (cost=0.00..14.00 rows=2 width=227) (actual time=0.005..0.005 rows=0 loops=1)
3	Filter: (eventdate = '2023-07-01'::date)
4	-> Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_2 (cost=0.00..14.00 rows=2 width=227) (actual time=0.001..0.001 rows=0 loop...
5	Filter: (eventdate = '2023-07-01'::date)
6	-> Seq Scan on sales_info_dp_2023_default sales_info_dp_3 (cost=0.00..14.00 rows=2 width=227) (actual time=0.001..0.001 rows=0 loops=1)
7	Filter: (eventdate = '2023-07-01'::date)
8	Planning Time: 0.118 ms
9	Execution Time: 0.027 ms

```
101
102  -- 5.3 Select an exact date
103  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate = '2023-07-01';
104
105  -- 5.4 Select an exact category
106  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE category = 'A';
107
108  -- 5.5 Select a list of categories
109  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE category IN ('A', 'B');
110
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..41026.40 rows=501490 width=11) (actual time=62.399..134.839 rows=500167 loops=1)
2	-> Seq Scan on sales_info_dp_2021_cat_a_b sales_info_dp_1 (cost=0.00..20587.99 rows=1 width=11) (actual time=62.275..62.275 rows=0 loops=1)
3	Filter: ((category)::text = 'A'::text)
4	Rows Removed by Filter: 998879
5	-> Seq Scan on sales_info_dp_2022_cat_a_b sales_info_dp_2 (cost=0.00..14.00 rows=2 width=227) (actual time=0.007..0.007 rows=0 loops=1)
6	Filter: ((category)::text = 'A'::text)
7	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_3 (cost=0.00..14.00 rows=2 width=227) (actual time=0.001..0.001 rows=0 loops=1)
8	Filter: ((category)::text = 'A'::text)
9	-> Seq Scan on sales_info_dp_2024_cat_a_b sales_info_dp_4 (cost=0.00..17902.96 rows=501485 width=11) (actual time=0.114..55.352 rows=500167 loop...
10	Filter: ((category)::text = 'A'::text)
11	Rows Removed by Filter: 499670
12	Planning Time: 0.157 ms
13	Execution Time: 143.944 ms

```
101
102  -- 5.3 Select an exact date
103  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE eventdate = '2023-07-01';
104
105  -- 5.4 Select an exact category
106  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE category = 'A';
107
108  -- 5.5 Select a list of categories
109  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE category IN ('A', 'B');
110
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..48512.56 rows=1998722 width=11) (actual time=0.013..272.499 rows=1998716 loops=1)
2	-> Seq Scan on sales_info_dp_2021_cat_a_b sales_info_dp_1 (cost=0.00..20587.99 rows=998879 width=11) (actual time=0.012..95.855 rows=998879 loop...
3	Filter: ((category)::text = ANY ('{A,B}'::text[]))
4	-> Seq Scan on sales_info_dp_2022_cat_a_b sales_info_dp_2 (cost=0.00..14.00 rows=3 width=227) (actual time=0.007..0.007 rows=0 loops=1)
5	Filter: ((category)::text = ANY ('{A,B}'::text[]))
6	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_3 (cost=0.00..14.00 rows=3 width=227) (actual time=0.001..0.001 rows=0 loops=1)
7	Filter: ((category)::text = ANY ('{A,B}'::text[]))
8	-> Seq Scan on sales_info_dp_2024_cat_a_b sales_info_dp_4 (cost=0.00..17902.96 rows=999837 width=11) (actual time=0.013..92.277 rows=999837 loop...
9	Filter: ((category)::text = ANY ('{A,B}'::text[]))
10	Planning Time: 0.192 ms
11	Execution Time: 316.973 ms

```
112  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE category IN ('A', 'B') AND eventdate = '2023-07-01';
113
114  -- 5.7 Count all rows
115  EXPLAIN ANALYZE SELECT COUNT(*) FROM SALES_INFO_DP;
116
117  -- 5.8 Count rows with a range of dates
118  EXPLAIN ANALYZE SELECT COUNT(*) FROM SALES_INFO_DP WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
119
120
121
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp (cost=0.00..14.80 rows=1 width=227) (actual time=0.005..0.005 rows=0 loop…
2	Filter: (((category)::text = ANY ('{A,B}'::text[])) AND (eventdate = '2023-07-01'::date))
3	Planning Time: 0.137 ms
4	Execution Time: 0.017 ms

```
112  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP WHERE category IN ('A', 'B') AND eventdate = '2023-07-01';
113
114  -- 5.7 Count all rows
115  EXPLAIN ANALYZE SELECT COUNT(*) FROM SALES_INFO_DP;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Finalize Aggregate (cost=130778.93..130778.94 rows=1 width=8) (actual time=493.684..497.245 rows=1 loops=1)
2	-> Gather (cost=130778.72..130778.93 rows=2 width=8) (actual time=493.527..497.224 rows=3 loops=1)
3	Workers Planned: 2
4	Workers Launched: 2
5	-> Partial Aggregate (cost=129778.72..129778.73 rows=1 width=8) (actual time=478.971..478.974 rows=1 loops=3)
6	-> Parallel Append (cost=0.00..119359.39 rows=4167732 width=0) (actual time=3.942..372.451 rows=3333333 loops=3)
7	-> Parallel Seq Scan on sales_info_dp_2021_cat_c_d_e_f sales_info_dp_2 (cost=0.00..38300.62 rows=1667162 width=0) (actual time=1.114..90.096 rows=1333729 loops…
8	-> Parallel Seq Scan on sales_info_dp_2024_cat_c_d_e_f sales_info_dp_11 (cost=0.00..38290.07 rows=1666707 width=0) (actual time=2.104..103.125 rows=2000048 loo…
9	-> Parallel Seq Scan on sales_info_dp_2021_cat_a_b sales_info_dp_1 (cost=0.00..12264.00 rows=416200 width=0) (actual time=0.032..72.485 rows=998879 loops=1)
10	-> Parallel Seq Scan on sales_info_dp_2024_cat_a_b sales_info_dp_10 (cost=0.00..9570.99 rows=416599 width=0) (actual time=4.381..77.931 rows=999837 loops=1)
11	-> Parallel Seq Scan on sales_info_dp_2021_default sales_info_dp_3 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
12	-> Parallel Seq Scan on sales_info_dp_2022_cat_a_b sales_info_dp_4 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
13	-> Parallel Seq Scan on sales_info_dp_2022_cat_c_d_e_f sales_info_dp_5 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
14	-> Parallel Seq Scan on sales_info_dp_2022_default sales_info_dp_6 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
15	-> Parallel Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_7 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
16	-> Parallel Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_8 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
17	-> Parallel Seq Scan on sales_info_dp_2023_default sales_info_dp_9 (cost=0.00..11.88 rows=188 width=0) (actual time=0.000..0.000 rows=0 loops=1)
18	-> Parallel Seq Scan on sales_info_dp_2024_default sales_info_dp_12 (cost=0.00..11.88 rows=188 width=0) (actual time=0.002..0.002 rows=0 loops=1)
19	Planning Time: 0.150 ms
20	JIT:
21	Functions: 44

Total rows: 24 of 24 Query complete 00:00:00.526 Ln 115, Col 1

```
117  -- 5.8 Count rows with a range of dates
118  EXPLAIN ANALYZE SELECT COUNT(*) FROM SALES_INFO_DP WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
119
120
121
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=44.45..44.46 rows=1 width=8) (actual time=0.008..0.009 rows=1 loops=1)
2	-> Append (cost=0.00..44.43 rows=6 width=0) (actual time=0.006..0.007 rows=0 loops=1)
3	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_1 (cost=0.00..14.80 rows=2 width=0) (actual time=0.004..0.004 rows=0 loops=1)
4	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
5	-> Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_2 (cost=0.00..14.80 rows=2 width=0) (actual time=0.001..0.001 rows=0 loop...
6	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
7	-> Seq Scan on sales_info_dp_2023_default sales_info_dp_3 (cost=0.00..14.80 rows=2 width=0) (actual time=0.001..0.001 rows=0 loops=1)
8	Filter: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
9	Planning Time: 0.140 ms
10	Execution Time: 0.032 ms

Task 3

```
126
127
128  SET max_parallel_workers_per_gather = 4;
129
130
131  EXPLAIN ANALYZE SELECT * FROM SALES_INFO;
132
133
134  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP;
135
136
137  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_SIMPLE;
138
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on sales_info_simple (cost=0.00..154056.75 rows=10000175 width=11) (actual time=0.060..455.260 rows=10000000 loop...
2	Planning Time: 0.034 ms
3	Execution Time: 637.447 ms

```
132
133
134 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP;
135
136
137 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_SIMPLE;
138
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..206877.40 rows=10002560 width=11) (actual time=0.011..882.144 rows=10000000 loops=1)
2	-> Seq Scan on sales_info_dp_2021_cat_a_b sales_info_dp_1 (cost=0.00..18090.79 rows=998879 width=11) (actual time=0.010..49.869 rows=998879 loops=1)
3	-> Seq Scan on sales_info_dp_2021_cat_c_d_e_f sales_info_dp_2 (cost=0.00..61640.88 rows=4001188 width=11) (actual time=0.025..183.265 rows=4001188 loops...
4	-> Seq Scan on sales_info_dp_2021_default sales_info_dp_3 (cost=0.00..13.20 rows=320 width=227) (actual time=0.007..0.007 rows=0 loops=1)
5	-> Seq Scan on sales_info_dp_2022_cat_a_b sales_info_dp_4 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
6	-> Seq Scan on sales_info_dp_2022_cat_c_d_e_f sales_info_dp_5 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.002 rows=0 loops=1)
7	-> Seq Scan on sales_info_dp_2022_default sales_info_dp_6 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
8	-> Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_7 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
9	-> Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_8 (cost=0.00..13.20 rows=320 width=227) (actual time=0.001..0.001 rows=0 loops=1)
10	-> Seq Scan on sales_info_dp_2023_default sales_info_dp_9 (cost=0.00..13.20 rows=320 width=227) (actual time=0.002..0.002 rows=0 loops=1)
11	-> Seq Scan on sales_info_dp_2024_cat_a_b sales_info_dp_10 (cost=0.00..15403.37 rows=999837 width=11) (actual time=0.012..43.708 rows=999837 loops=1)
12	-> Seq Scan on sales_info_dp_2024_cat_c_d_e_f sales_info_dp_11 (cost=0.00..61623.96 rows=4000096 width=11) (actual time=0.016..206.827 rows=4000096 loop...
13	-> Seq Scan on sales_info_dp_2024_default sales_info_dp_12 (cost=0.00..13.20 rows=320 width=227) (actual time=0.009..0.009 rows=0 loops=1)
14	Planning Time: 0.158 ms

Successfully run

```
130
131 EXPLAIN ANALYZE SELECT * FROM SALES_INFO;
132
133
134 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP;
135
136
137 EXPLAIN ANALYZE SELECT * FROM SALES_INFO_SIMPLE;
138
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=0.00..102204.74 rows=5008649 width=11) (actual time=0.211..473.877 rows=5000794 loops=1)
2	-> Seq Scan on sales_info_2022 sales_info_1 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.004..0.005 rows=0 loops=1)
3	-> Seq Scan on sales_info_2023 sales_info_2 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.001..0.001 rows=0 loops=1)
4	-> Seq Scan on sales_info_2024 sales_info_3 (cost=0.00..77049.09 rows=5001409 width=11) (actual time=0.205..257.946 rows=5000794 loop...
5	-> Seq Scan on sales_info_2025 sales_info_4 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.005..0.005 rows=0 loops=1)
6	-> Seq Scan on sales_info_2026 sales_info_5 (cost=0.00..28.10 rows=1810 width=17) (actual time=0.003..0.004 rows=0 loops=1)
7	Planning Time: 0.082 ms
8	Execution Time: 586.367 ms

```
135
136
137  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_SIMPLE;
138
139
140
141  EXPLAIN ANALYZE SELECT * FROM SALES_INFO ORDER BY eventdate;
142
143  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_DP ORDER BY eventdate;
144
145  EXPLAIN ANALYZE SELECT * FROM SALES_INFO_SIMPLE ORDER BY eventdate;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Append (cost=1.04..144307.72 rows=5008034 width=11) (actual time=0.050..617.866 rows=5000794 loops=1)
2	-> Index Scan using sales_info_2022_eventdate_idx on sales_info_2022 sales_info_1 (cost=0.15..71.30 rows=1810 width=17) (actual time=0.005..0.005 rows=0 loops=1
3	-> Index Scan using sales_info_2023_eventdate_idx on sales_info_2023 sales_info_2 (cost=0.15..71.30 rows=1810 width=17) (actual time=0.002..0.002 rows=0 loops=1
4	-> Index Scan using sales_info_2024_eventdate_idx on sales_info_2024 sales_info_3 (cost=0.43..118982.34 rows=5000794 width=11) (actual time=0.043..417.184 rows
5	-> Index Scan using sales_info_2025_eventdate_idx on sales_info_2025 sales_info_4 (cost=0.15..71.30 rows=1810 width=17) (actual time=0.008..0.008 rows=0 loops=1
6	-> Index Scan using sales_info_2026_eventdate_idx on sales_info_2026 sales_info_5 (cost=0.15..71.30 rows=1810 width=17) (actual time=0.004..0.004 rows=0 loops=1
7	Planning Time: 0.159 ms
8	Execution Time: 723.178 ms

S

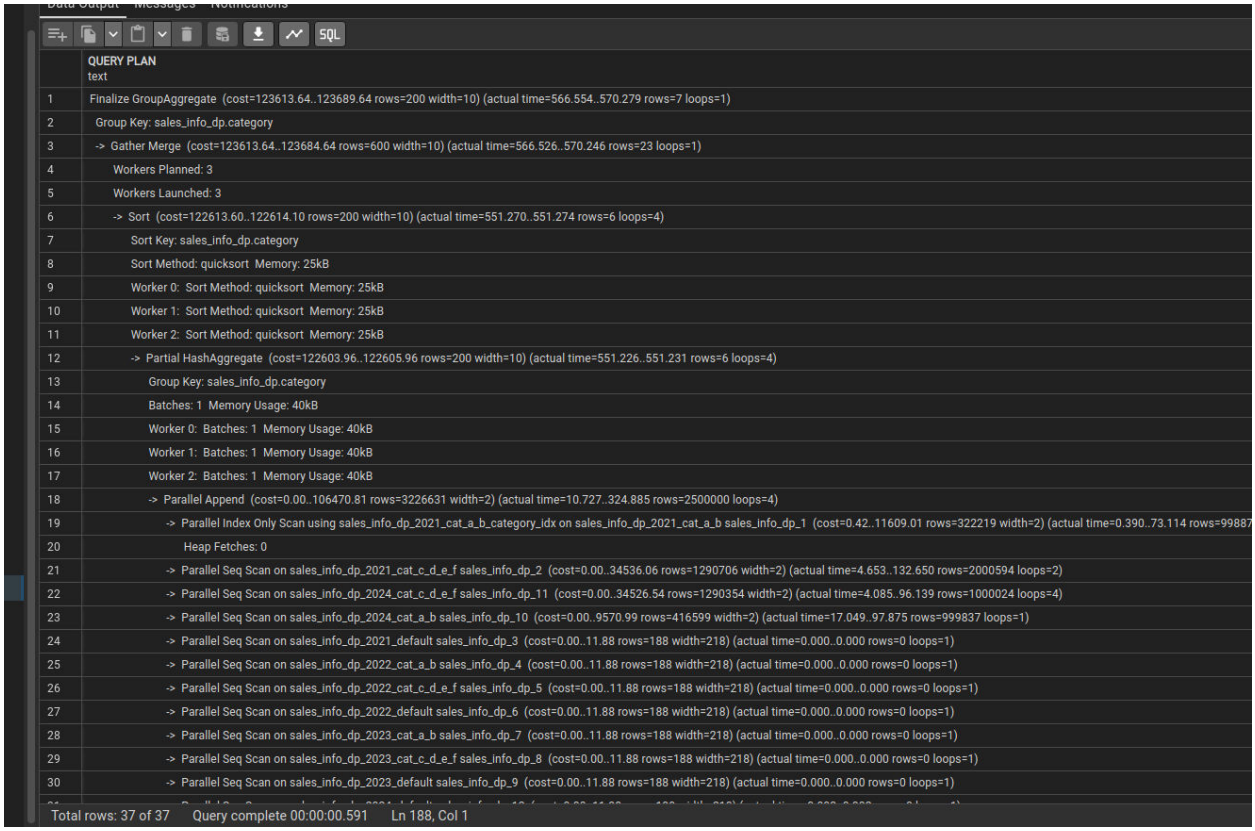
```
152
153
154  EXPLAIN ANALYZE SELECT COUNT(*) FROM SALES_INFO_SIMPLE;
155
156
157  EXPLAIN ANALYZE SELECT * FROM SALES_INFO WHERE eventdate BETWEEN '2023-01-01' AND '2023-12-31';
158
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Bitmap Heap Scan on sales_info_2023 sales_info (cost=4.24..14.81 rows=9 width=17) (actual time=0.005..0.005 rows=0 loops=1)
2	Recheck Cond: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
3	-> Bitmap Index Scan on sales_info_2023_eventdate_idx (cost=0.00..4.24 rows=9 width=0) (actual time=0.003..0.003 rows=0 loop...
4	Index Cond: ((eventdate >= '2023-01-01'::date) AND (eventdate <= '2023-12-31'::date))
5	Planning Time: 0.095 ms
6	Execution Time: 0.016 ms

Total rows: 6 of 6 Query complete 00:00:00.045 Ln 158, Col 1

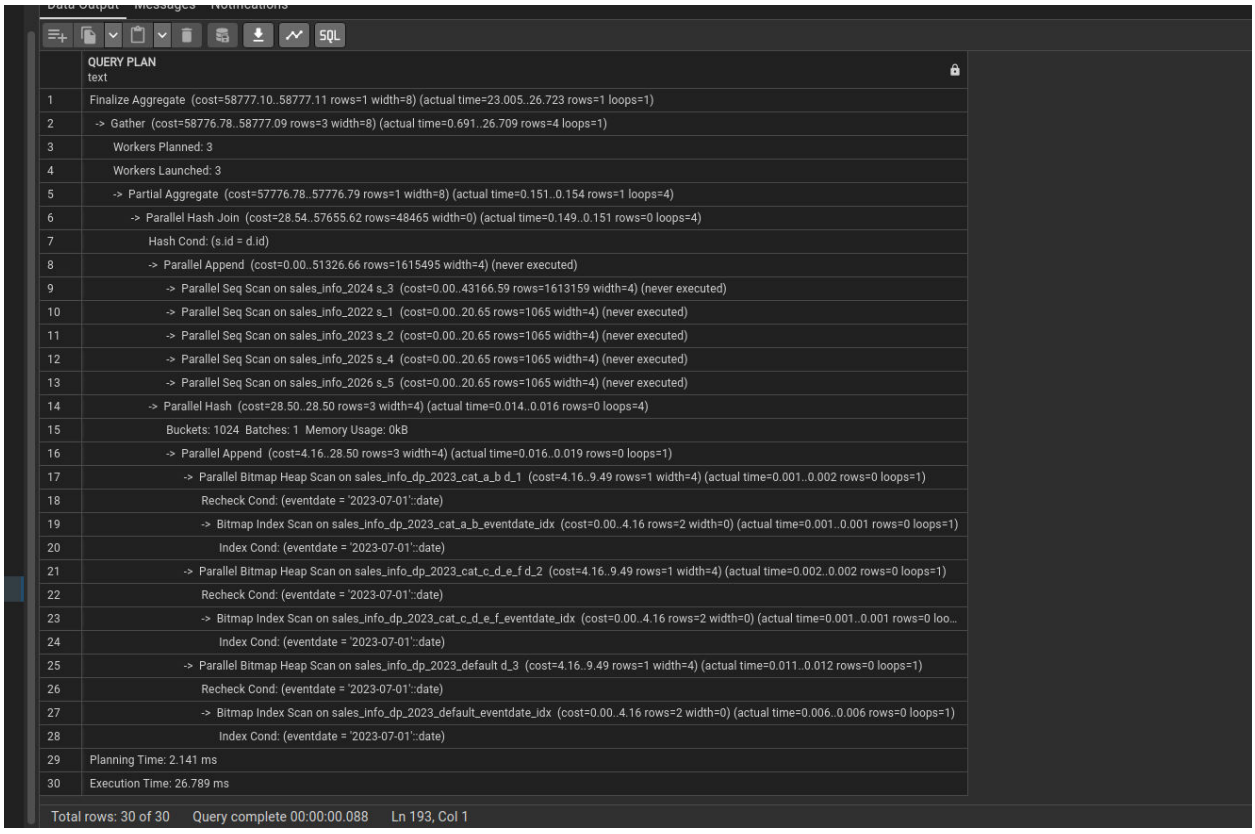
S



	QUERY PLAN text
1	Finalize GroupAggregate (cost=123613.64..123689.64 rows=200 width=10) (actual time=566.554..570.279 rows=7 loops=1)
2	Group Key: sales_info_dp.category
3	-> Gather Merge (cost=123613.64..123684.64 rows=600 width=10) (actual time=566.526..570.246 rows=23 loops=1)
4	Workers Planned: 3
5	Workers Launched: 3
6	-> Sort (cost=122613.60..122614.10 rows=200 width=10) (actual time=551.270..551.274 rows=6 loops=4)
7	Sort Key: sales_info_dp.category
8	Sort Method: quicksort Memory: 25kB
9	Worker 0: Sort Method: quicksort Memory: 25kB
10	Worker 1: Sort Method: quicksort Memory: 25kB
11	Worker 2: Sort Method: quicksort Memory: 25kB
12	-> Partial HashAggregate (cost=122603.96..122605.96 rows=200 width=10) (actual time=551.226..551.231 rows=6 loops=4)
13	Group Key: sales_info_dp.category
14	Batches: 1 Memory Usage: 40kB
15	Worker 0: Batches: 1 Memory Usage: 40kB
16	Worker 1: Batches: 1 Memory Usage: 40kB
17	Worker 2: Batches: 1 Memory Usage: 40kB
18	-> Parallel Append (cost=0.00..106470.81 rows=3226631 width=2) (actual time=10.727..324.885 rows=2500000 loops=4)
19	-> Parallel Index Only Scan using sales_info_dp_2021_cat_a_b_category_idx on sales_info_dp_2021_cat_a_b sales_info_dp_1 (cost=0.42..11609.01 rows=322219 width=2) (actual time=0.390..73.114 rows=99887
20	Heap Fetches: 0
21	-> Parallel Seq Scan on sales_info_dp_2021_cat_c_d_e_f sales_info_dp_2 (cost=0.00..34536.06 rows=1290706 width=2) (actual time=4.653..132.650 rows=2000594 loops=2)
22	-> Parallel Seq Scan on sales_info_dp_2024_cat_c_d_e_f sales_info_dp_11 (cost=0.00..34526.54 rows=1290354 width=2) (actual time=4.085..96.139 rows=1000024 loops=4)
23	-> Parallel Seq Scan on sales_info_dp_2024_cat_a_b sales_info_dp_10 (cost=0.00..9570.99 rows=416599 width=2) (actual time=17.049..97.875 rows=999837 loops=1)
24	-> Parallel Seq Scan on sales_info_dp_2021_default sales_info_dp_3 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)
25	-> Parallel Seq Scan on sales_info_dp_2022_cat_a_b sales_info_dp_4 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)
26	-> Parallel Seq Scan on sales_info_dp_2022_cat_c_d_e_f sales_info_dp_5 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)
27	-> Parallel Seq Scan on sales_info_dp_2022_default sales_info_dp_6 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)
28	-> Parallel Seq Scan on sales_info_dp_2023_cat_a_b sales_info_dp_7 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)
29	-> Parallel Seq Scan on sales_info_dp_2023_cat_c_d_e_f sales_info_dp_8 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)
30	-> Parallel Seq Scan on sales_info_dp_2023_default sales_info_dp_9 (cost=0.00..11.88 rows=188 width=218) (actual time=0.000..0.000 rows=0 loops=1)

Total rows: 37 of 37 Query complete 00:00:00.591 Ln 188, Col 1

Selecting count of row , as wee see it’s goes through all partitions in parrallel after creating index.



	QUERY PLAN text
1	Finalize Aggregate (cost=58777.10..58777.11 rows=1 width=8) (actual time=23.005..26.723 rows=1 loops=1)
2	-> Gather (cost=58776.78..58777.09 rows=3 width=8) (actual time=0.691..26.709 rows=4 loops=1)
3	Workers Planned: 3
4	Workers Launched: 3
5	-> Partial Aggregate (cost=57776.78..57776.79 rows=1 width=8) (actual time=0.151..0.154 rows=1 loops=4)
6	-> Parallel Hash Join (cost=28.54..57655.62 rows=48465 width=0) (actual time=0.149..0.151 rows=0 loops=4)
7	Hash Cond: (s.id = d.id)
8	-> Parallel Append (cost=0.00..51326.66 rows=1615495 width=4) (never executed)
9	-> Parallel Seq Scan on sales_info_2024 s_3 (cost=0.00..43166.59 rows=1613159 width=4) (never executed)
10	-> Parallel Seq Scan on sales_info_2022 s_1 (cost=0.00..20.65 rows=1065 width=4) (never executed)
11	-> Parallel Seq Scan on sales_info_2023 s_2 (cost=0.00..20.65 rows=1065 width=4) (never executed)
12	-> Parallel Seq Scan on sales_info_2025 s_4 (cost=0.00..20.65 rows=1065 width=4) (never executed)
13	-> Parallel Seq Scan on sales_info_2026 s_5 (cost=0.00..20.65 rows=1065 width=4) (never executed)
14	-> Parallel Hash (cost=28.50..28.50 rows=3 width=4) (actual time=0.014..0.016 rows=0 loops=4)
15	Buckets: 1024 Batches: 1 Memory Usage: 0kB
16	-> Parallel Append (cost=4.16..28.50 rows=3 width=4) (actual time=0.016..0.019 rows=0 loops=1)
17	-> Parallel Bitmap Heap Scan on sales_info_dp_2023_cat_a_b d_1 (cost=4.16..9.49 rows=1 width=4) (actual time=0.001..0.002 rows=0 loops=1)
18	Recheck Cond: (eventdate = '2023-07-01'::date)
19	-> Bitmap Index Scan on sales_info_dp_2023_cat_a_b_eventdate_idx (cost=0.00..4.16 rows=2 width=0) (actual time=0.001..0.001 rows=0 loops=1)
20	Index Cond: (eventdate = '2023-07-01'::date)
21	-> Parallel Bitmap Heap Scan on sales_info_dp_2023_cat_c_d_e_f d_2 (cost=4.16..9.49 rows=1 width=4) (actual time=0.002..0.002 rows=0 loops=1)
22	Recheck Cond: (eventdate = '2023-07-01'::date)
23	-> Bitmap Index Scan on sales_info_dp_2023_cat_c_d_e_f_eventdate_idx (cost=0.00..4.16 rows=2 width=0) (actual time=0.001..0.001 rows=0 loo...
24	Index Cond: (eventdate = '2023-07-01'::date)
25	-> Parallel Bitmap Heap Scan on sales_info_dp_2023_default d_3 (cost=4.16..9.49 rows=1 width=4) (actual time=0.011..0.012 rows=0 loops=1)
26	Recheck Cond: (eventdate = '2023-07-01'::date)
27	-> Bitmap Index Scan on sales_info_dp_2023_default_eventdate_idx (cost=0.00..4.16 rows=2 width=0) (actual time=0.006..0.006 rows=0 loops=1)
28	Index Cond: (eventdate = '2023-07-01'::date)
29	Planning Time: 2.141 ms
30	Execution Time: 26.789 ms

Total rows: 30 of 30 Query complete 00:00:00.088 Ln 193, Col 1