IS 497 - Assignment T4B

Team 1: Jigyashu Mukeshkumar Saravta, Nitya Tirumalasetti, Sanjanaa Sridhar, Sundar Lakshmi Narasimhan

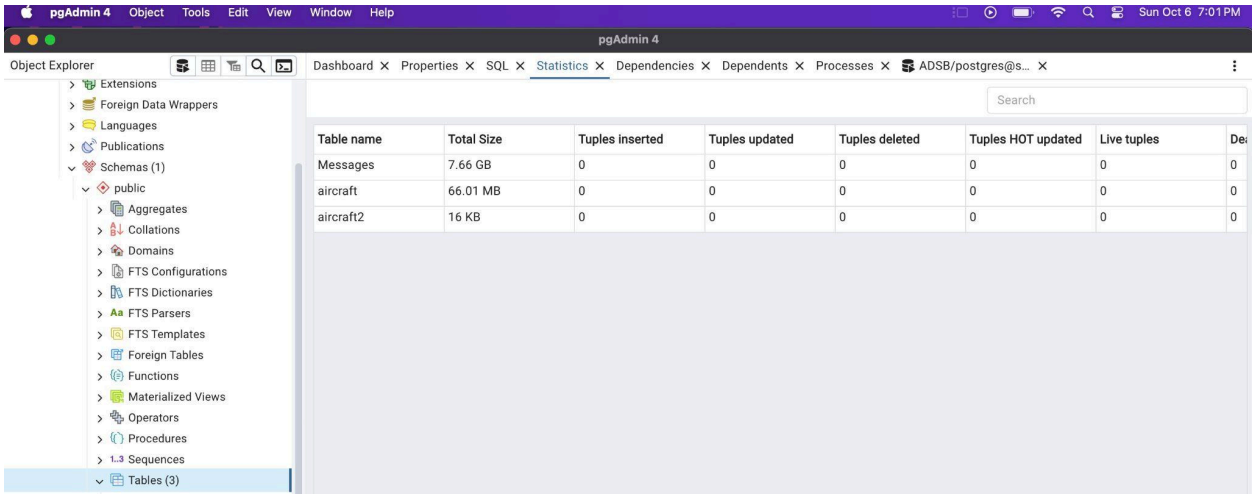
---

Import Times

Team Member	Timeline	Dataset	Import Time (s)	Network	Speed (MB/sec)
Sundar	12/30/2023 - 02/05/2024 (no data for Jan was available)	Messages_1	408	Campus Network (IllinoisNet)	2.26
		Messages_2	418		
		Messages_3	395		
		Messages_4	431		
		Messages_5	393		
		Messages_6	232		
		Messages_7	86		
Nitya	02/01/2024 - 02/07/2024	Messages_1	112.02	Home Network (VOLO)	7.11
		Messages_2	117.22		
		Messages_3	97.24		
		Messages_4	108.46		
		Messages_5	104.82		
		Messages_6	93.11		
		Messages_7	113.7		
Jigyashu	12/30/2023 - 02/05/2024 (no data for Jan was available)	Messages_1	236	Home Network (VOLO)	1.84
		Messages_2	350		
		Messages_3	478		
		Messages_4	492		

		Messages_5	434		
		Messages_6	460		
		Messages_7	452		
Sanjanaa	02/01/2024 - 02/07/2024	Messages_1	78	Campus Network (IllinoisNet)	8.71
		Messages_2	67		
		Messages_3	57		
		Messages_4	92		
		Messages_5	84		
		Messages_6	68		
		Messages_7	76		

Table Statistics



Please ignore aircraft2

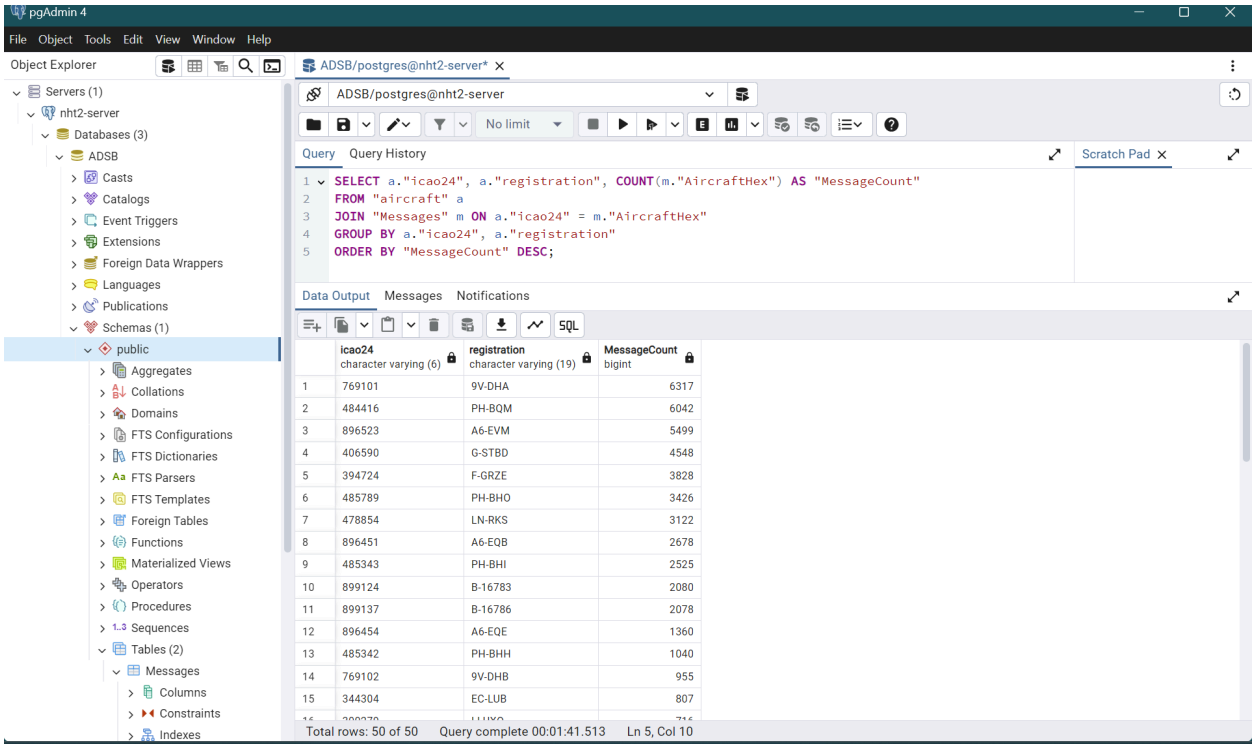
Queries

1. We tried a query that returns a list of aircraft, their registration numbers, and how many messages are associated with each aircraft in the Messages table. The results are ordered by the number of messages, so the aircraft with the highest number of messages appears first.

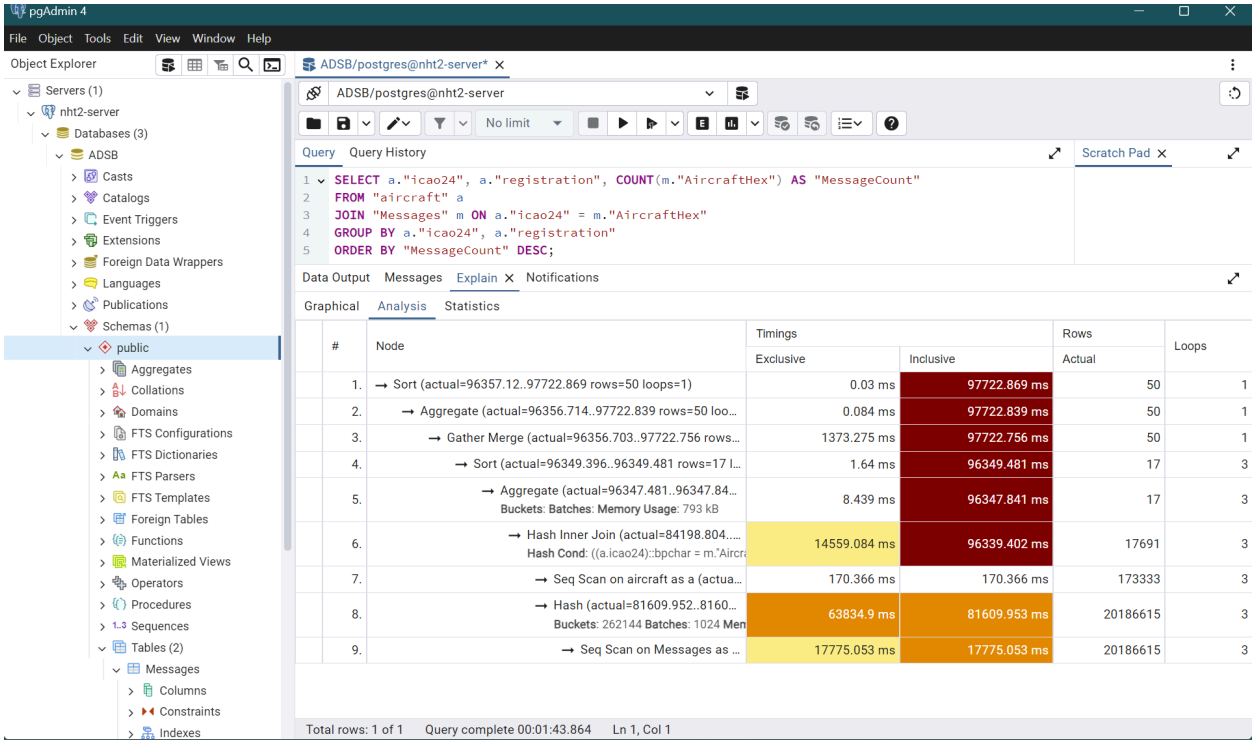
Query

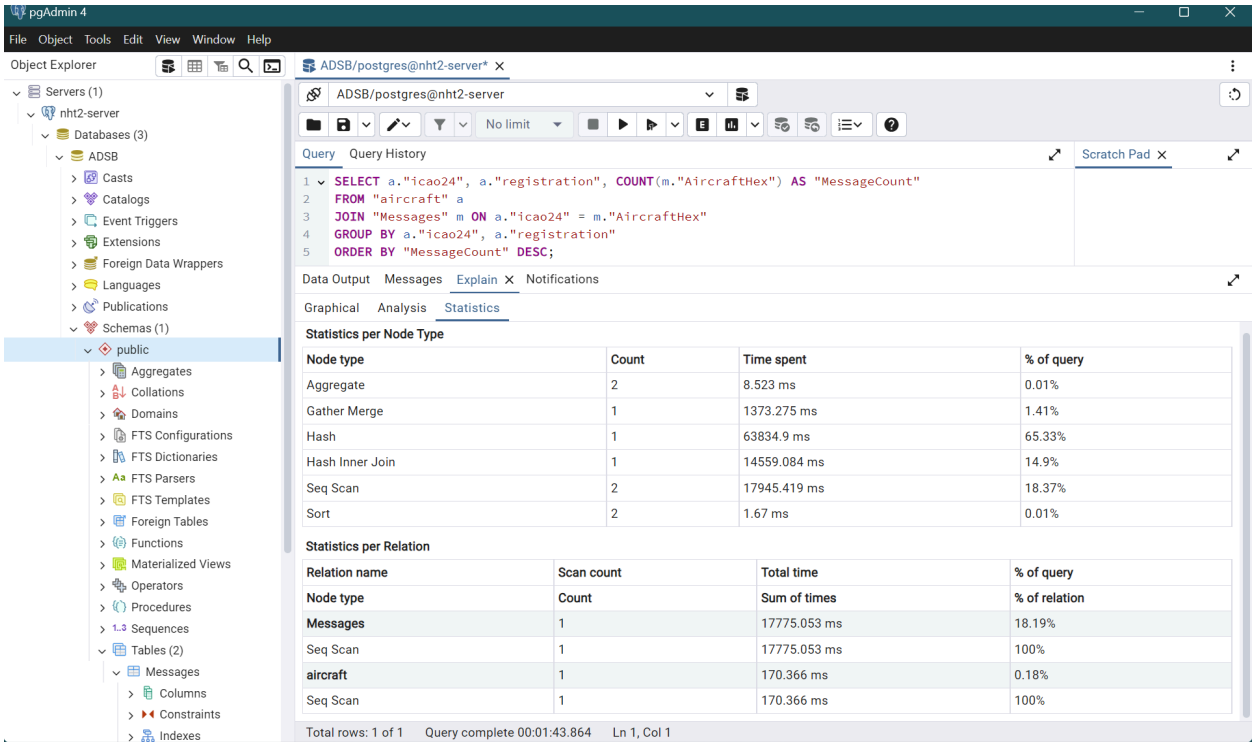
```
SELECT a."icao24", a."registration", COUNT(m."AircraftHex") AS
"MessageCount"
FROM "aircraft" a
JOIN "Messages" m ON a."icao24" = m."AircraftHex"
GROUP BY  a."icao24", a."registration"
ORDER BY "MessageCount" DESC;
```

- Original Query - took 1 minute 41 sections to execute

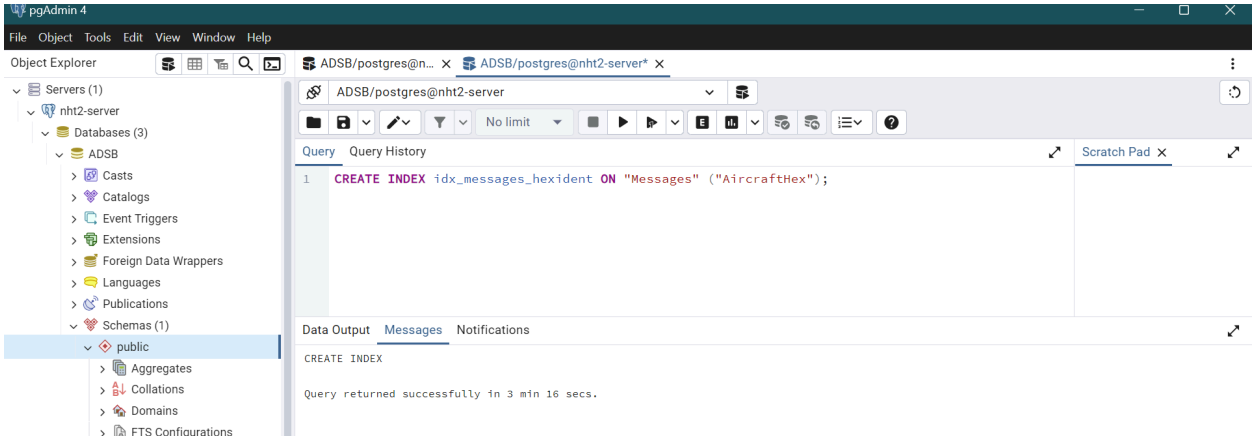


- Ran the explain and analyze statement with Timing filter

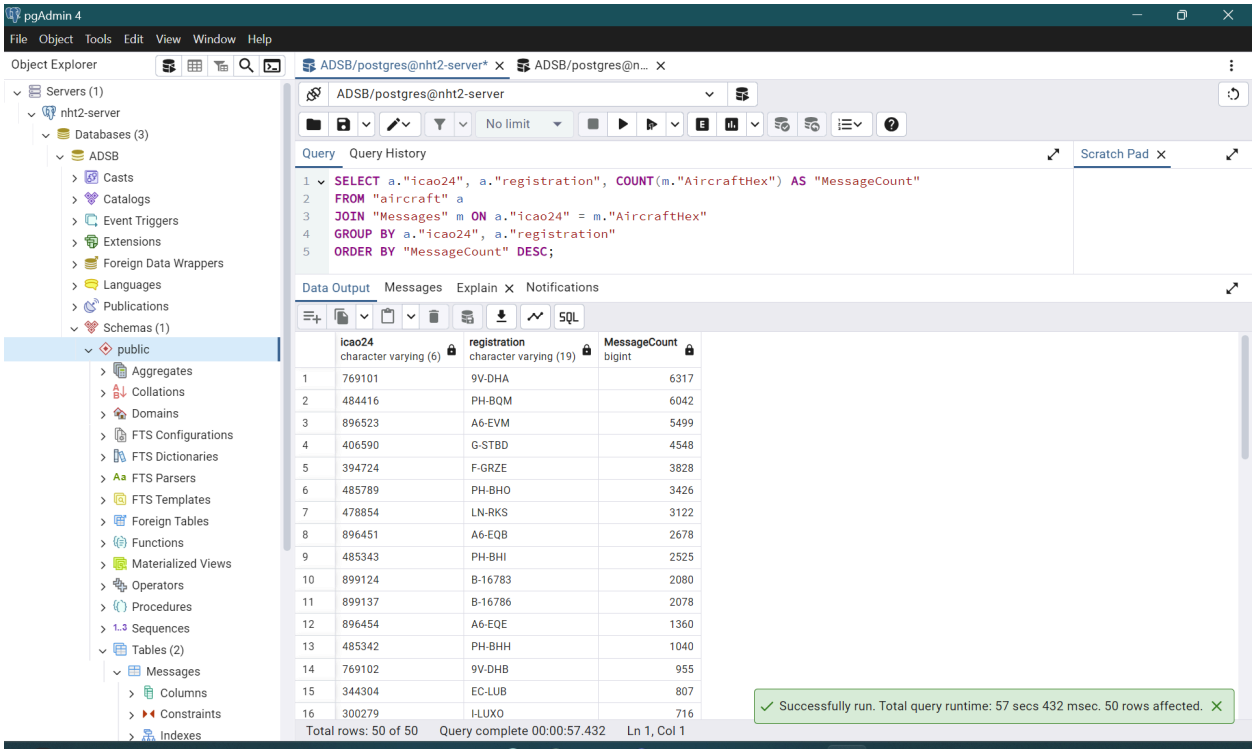




Based on the above results, it is clear that a major portion of the query is composed of the hash and sequential scan functions. Messages table has millions of rows and performing sequential scans over such massive data can slow down the query processing. Hence, we added an index to AircraftHex column in Messages table to quickly locate matching values rather than scanning the entire table.



Re-ran the query and noticed that the execution time reduced to 57s from 1m 41s

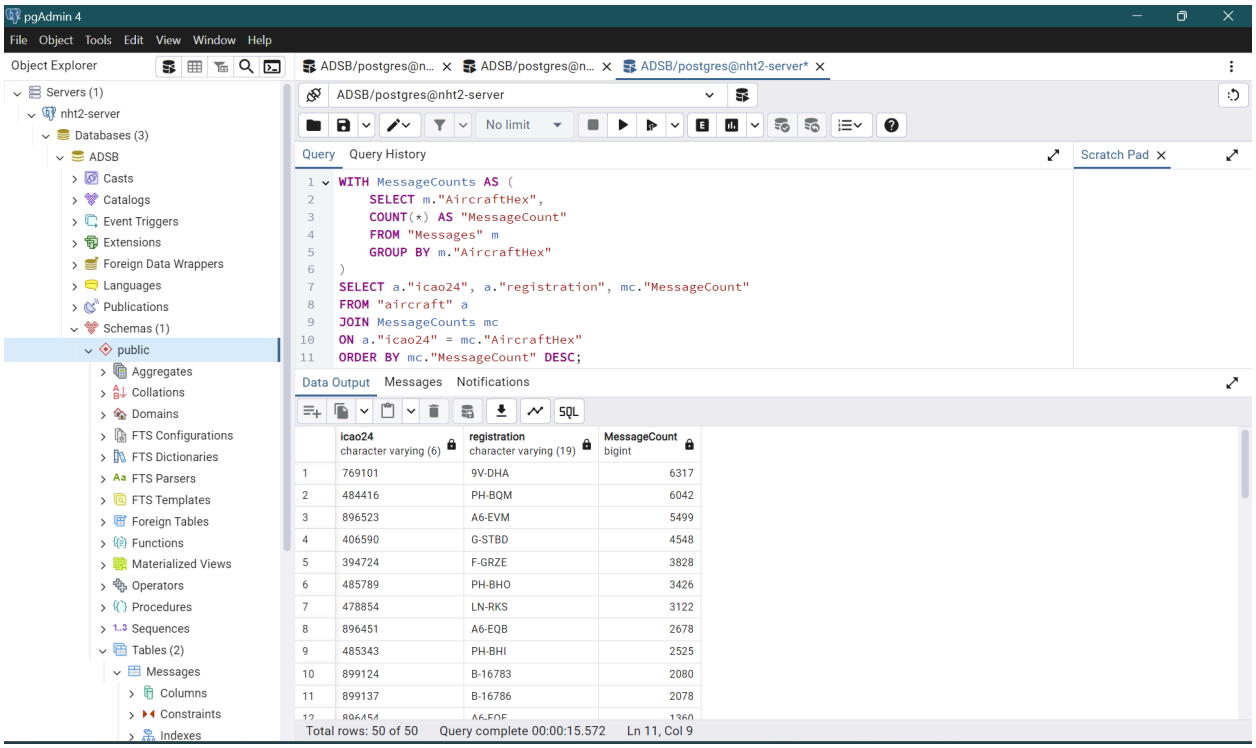


Query with CTE

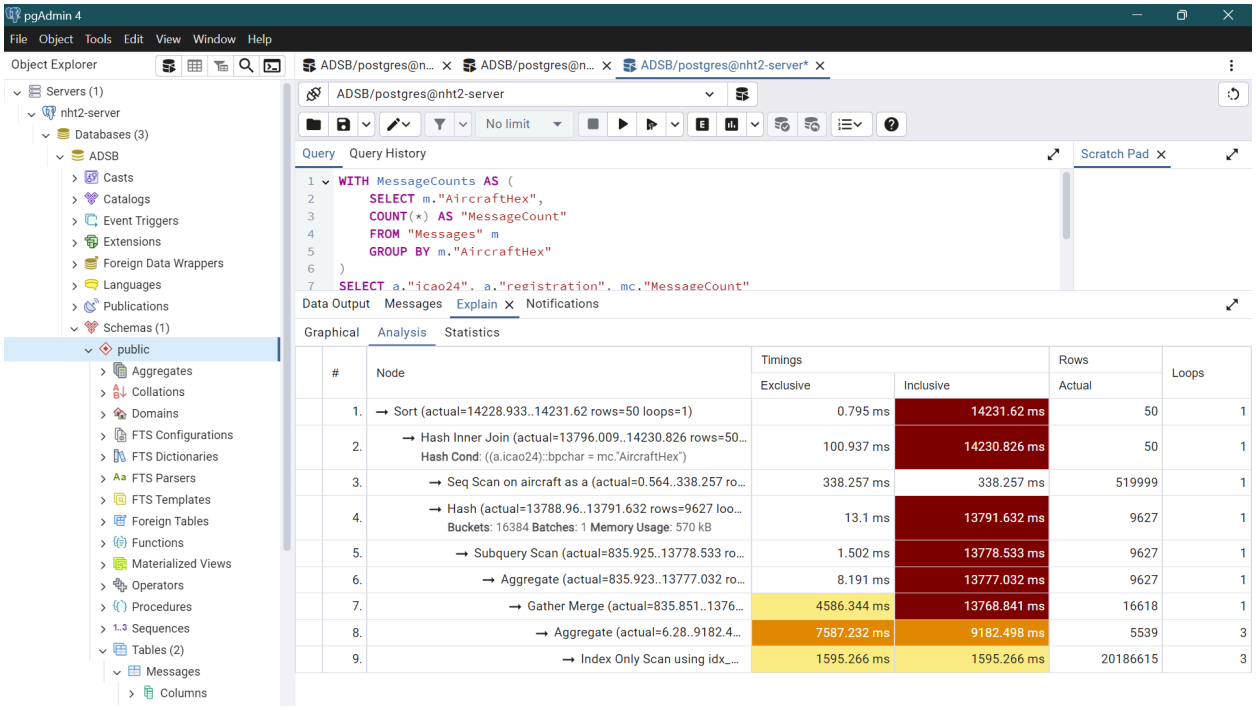
```
WITH MessageCounts AS (
    SELECT    m."AircraftHex", COUNT(*) AS "MessageCount"
    FROM "Messages" m
    GROUP BY m."AircraftHex"
)
SELECT  a."icao24",  a."registration",  mc."MessageCount"
FROM  "aircraft" a
JOIN  MessageCounts mc ON a."icao24" = mc."AircraftHex"
```

```
ORDER BY mc."MessageCount" DESC;
```

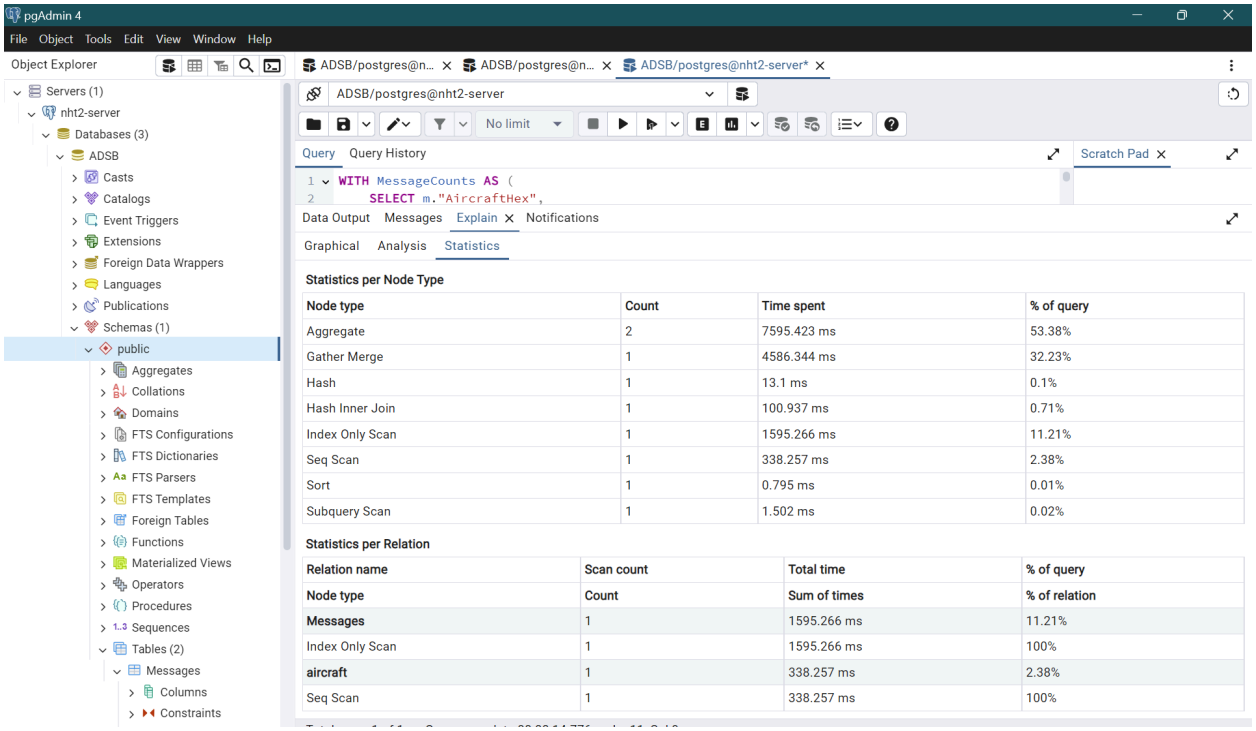
Further implemented Common Table Expression approach to avoid redundant iterations during the join operation for keeping track of the messages count. This, along with indexing, helped in reducing the query execution time to 15s.



Explain and Analyze results for query with index and CTE implementation. It can be observed that there is a shift from Sequential Scan to Index Scan in this latest query which reduces the processing times significantly. Also, CTE leads to pre-aggregation of data (aggregate node type comprises 53.38% of the query) which results in join operation being applied to a smaller and filtered dataset thereby making it faster compared to the original query's hash join operation.



#	Node	Timings		Rows	Loops
		Exclusive	Inclusive	Actual	
1.	→ Sort (actual=14228.933..14231.62 rows=50 loops=1)	0.795 ms	14231.62 ms	50	1
2.	→ Hash Inner Join (actual=13796.009..14230.826 rows=50... **Hash Cond**: ((a.icao24)::bpchar = mc."AircraftHex")	100.937 ms	14230.826 ms	50	1
3.	→ Seq Scan on aircraft as a (actual=0.564..338.257 ro...	338.257 ms	338.257 ms	519999	1
4.	→ Hash (actual=13788.96..13791.632 rows=9627 loo... **Buckets**: 16384 **Batches**: 1 **Memory Usage**: 570 kB	13.1 ms	13791.632 ms	9627	1
5.	→ Subquery Scan (actual=835.925..13778.533 ro...	1.502 ms	13778.533 ms	9627	1
6.	→ Aggregate (actual=835.923..13777.032 ro...	8.191 ms	13777.032 ms	9627	1
7.	→ Gather Merge (actual=835.851..1376...	4586.344 ms	13768.841 ms	16618	1
8.	→ Aggregate (actual=6.28..9182.4...	7587.232 ms	9182.498 ms	5539	3
9.	→ Index Only Scan using idx_...	1595.266 ms	1595.266 ms	20186615	3



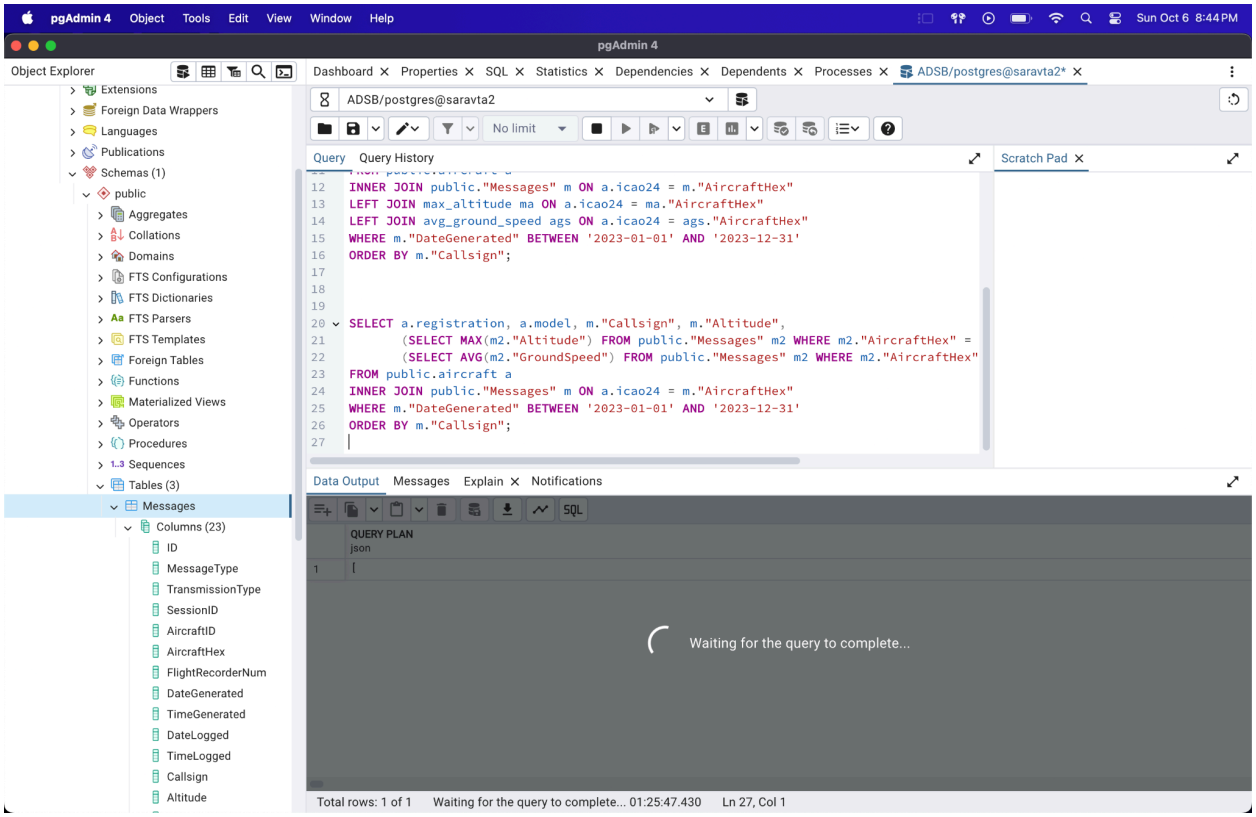
Statistics per Node Type

Node type	Count	Time spent	% of query
Aggregate	2	7595.423 ms	53.38%
Gather Merge	1	4586.344 ms	32.23%
Hash	1	13.1 ms	0.1%
Hash Inner Join	1	100.937 ms	0.71%
Index Only Scan	1	1595.266 ms	11.21%
Seq Scan	1	338.257 ms	2.38%
Sort	1	0.795 ms	0.01%
Subquery Scan	1	1.502 ms	0.02%

Statistics per Relation

Relation name	Scan count	Total time	% of query
Node type	**Count**	**Sum of times**	**% of relation**
Messages	1	1595.266 ms	11.21%
Index Only Scan	1	1595.266 ms	100%
aircraft	1	338.257 ms	2.38%
Seq Scan	1	338.257 ms	100%

2. With curiosity about just how efficient CTEs prove to be, we tried optimizing a query with 2 subqueries and 1 INNER JOIN with a CTE. This query is to return aircraft and message information for a specific time period and to return the maximum altitude and average ground speed of each aircraft.



With a CTE, we observed a large time reduction reducing the execution time to just 5m from 80+ mins.

Query | Query History

```sql
WITH max_altitude AS (
    SELECT "AircraftHex", MAX("Altitude") AS max_alt
    FROM public."Messages"
    GROUP BY "AircraftHex"
), avg_ground_speed AS (
    SELECT "AircraftHex", AVG("GroundSpeed") AS avg_speed
    FROM public."Messages"
    GROUP BY "AircraftHex"
)
SELECT a.registration, a.model, m."Callsign", m."Altitude", ma.max_alt, ags.avg_speed
FROM public.aircraft a
INNER JOIN public."Messages" m ON a.icao24 = m."AircraftHex"
LEFT JOIN max_altitude ma ON a.icao24 = ma."AircraftHex"
LEFT JOIN avg_ground_speed ags ON a.icao24 = ags."AircraftHex"
WHERE m."DateGenerated" BETWEEN '2023-01-01' AND '2023-12-31'
ORDER BY m."Callsign";
```

Data Output | Messages | Explain | Notifications

SQL

	registration character varying	model character varying (93)	Callsign character varying (20)	Altitude integer	max_alt integer	avg_speed numeric
1	G-VIIL	BOEING 777-236	BAW24R	[null]	34875	405.4285714285714286
2	G-VIIL	BOEING 777-236	BAW24R	[null]	34875	405.4285714285714286
3	G-VIIL	BOEING 777-236	BAW24R	[null]	34875	405.4285714285714286
4	G-VIIL	BOEING 777-236	BAW24R	[null]	34875	405.4285714285714286
5	G-VIIL	BOEING 777-236	BAW24R	[null]	34875	405.4285714285714286
6	G-VIIL	BOEING 777-236	BAW24R	[null]	34875	405.4285714285714286
7	A6-XWA	A350-1041	ETD76T	[null]	25850	389.5833333333333333
8	ET-ASI	B787-8	ETH575	[null]	27225	423.0000000000000000
9	EC-NLP	A350-941	IBE6170	[null]	39000	546.1156462585034014
10	EC-NLP	A350-941	IBE6170	[null]	39000	546.1156462585034014
11	EC-NLP	A350-941	IBE6170	[null]	39000	546.1156462585034014

Total rows: 1000 of 31214 Query complete 00:05:01.651 Ln 3, Col 27

Graphical Analysis Statistics

Statistics per Node Type

Node type	Count	Time spent	% of query
Aggregate	2	214733.602 ms	60.91%
Gather Merge	1	552.427 ms	0.16%
Hash	3	42462.009 ms	12.05%
Hash Inner Join	1	4954.945 ms	1.41%
Hash Left Join	2	16.355 ms	0.01%
Seq Scan	4	89856.82 ms	25.49%
Sort	1	10.385 ms	0.01%
Subquery Scan	2	8.13 ms	0.01%

Statistics per Relation

Relation name	Scan count	Total time	% of query
Node type	**Count**	**Sum of times**	**% of relation**
Messages	3	89732.17 ms	25.45%
Seq Scan	3	89732.17 ms	100%
aircraft	1	124.65 ms	0.04%
Seq Scan	1	124.65 ms	100%

Data Output Messages Explain ✕ Notifications

Graphical Analysis Statistics

#	Node	Timings		Rows	Loops
		Exclusive	Inclusive	Actual	
1.	→ Gather Merge (actual=352048.509..352594.67 rows=3121…	552.427 ms	352594.67 ms	31214	1
2.	→ Sort (actual=352039.633..352042.243 rows=10405 l…	10.385 ms	352042.243 ms	10405	3
3.	→ Hash Left Join (actual=348108.98..352031.859 r… **Hash Cond**: (a.icao24 = ags."AircraftHex")	5.612 ms	352031.859 ms	10405	3
4.	→ Hash Left Join (actual=202820.987..20674… **Hash Cond**: (a.icao24 = ma."AircraftHex")	10.743 ms	206740.096 ms	10405	3
5.	→ Hash Inner Join (actual=60286.746..64… **Hash Cond**: (a.icao24 = m."AircraftHex")	4954.945 ms	64200.15 ms	10405	3
6.	→ Seq Scan on aircraft as a (actual=…	124.65 ms	124.65 ms	173333	3
7.	→ Hash (actual=59120.554..59120.5… **Buckets**: 131072 **Batches**: 256 **Memory**	42454.155 ms	59120.555 ms	5356501	3
8.	→ Seq Scan on Messages as m … **Filter**: (("DateGenerated" >= '2023- **Rows Removed by Filter**: 1446822	16666.4 ms	16666.4 ms	5356501	3
9.	→ Hash (actual=142529.2..142529.203 r… **Buckets**: 16384 **Batches**: 1 **Memory Usage**: 5	4.067 ms	142529.203 ms	9630	3
10.	→ Subquery Scan (actual=142516.4…	6.724 ms	142525.137 ms	9630	3
11.	→ Aggregate (actual=142514.8… **Buckets**: **Batches**: **Memory Usage**:	106381.389 ms	142518.413 ms	9630	3
12.	→ Seq Scan on Messages …	36137.024 ms	36137.024 ms	59474190	3
13.	→ Hash (actual=145286.15..145286.151 rows… **Buckets**: 16384 **Batches**: 1 **Memory Usage**: 607 kB	3.787 ms	145286.151 ms	9630	3
14.	→ Subquery Scan (actual=145272.38..14…	1.406 ms	145282.365 ms	9630	3
15.	→ Aggregate (actual=145272.374..1… **Buckets**: **Batches**: **Memory Usage**: 1681	108352.213 ms	145280.959 ms	9630	3
16.	→ Seq Scan on Messages as M…	36928.746 ms	36928.746 ms	59474190	3