CS 587 Database Implementation
Winter 2021
Database Benchmarking Project - Part 3

Team

Sai Deepika Gooty Agraharam | Swetha Venkatesan

Single System

Evaluate a single relational system PostgreSQL by changing system parameters and varying relation size.

PostgreSQL

- Open Source
- Robust / Scaling
- Immense Flexibility (Parameters and Optimizer options)
- Prior Knowledge / Rich Documentation / Interactive Community

Objectives

- Design and implement and run a database benchmark
- Implementation of Join algorithm in PostgreSQL
- Impact of index scan and selectivity on the execution time
- Variation of execution time by varying memory usage

Implementation

SYSTEM CONFIGURATION

Local Machine

- 16GB RAM
- Windows operating system
- Postgres with Psql version 13.1

Google Cloud Platform VM

- 2GB RAM
- Ubuntu operating system
- Postgres with Psql version 12.6

DATASETS

- ONEKTUP - 1000
- TENKTUP - 10,000
- FIFTYKTUP - 50,000
- HUNDREDKTUP - 100,000
- MILLIONTUP - 1,000,000

EXPERIMENTS

IN

LOCAL SYSTEM

Experiment -1 Testing the 10% rule of thumb

1% - select count(*) from milliontup where unique2 between 792 and 10791;

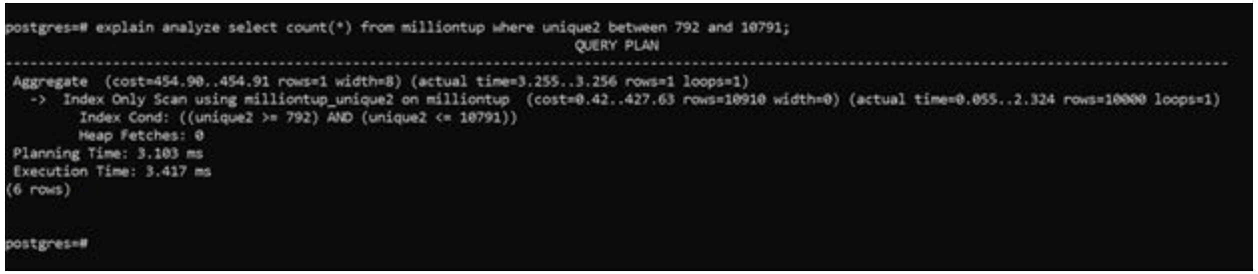
5% - select count(*) from milliontup where unique2 between 792 and 50792;

10% - explain analyze select count(*) from milliontup where unique2 between 792 and 100791;

Experiment -1 Testing the 10% rule of thumb

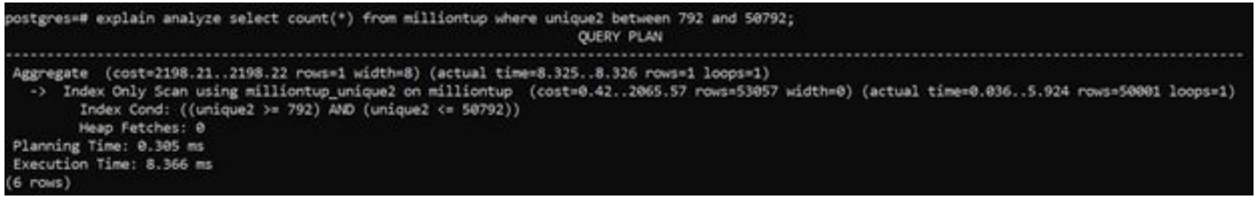
With No Index

1%



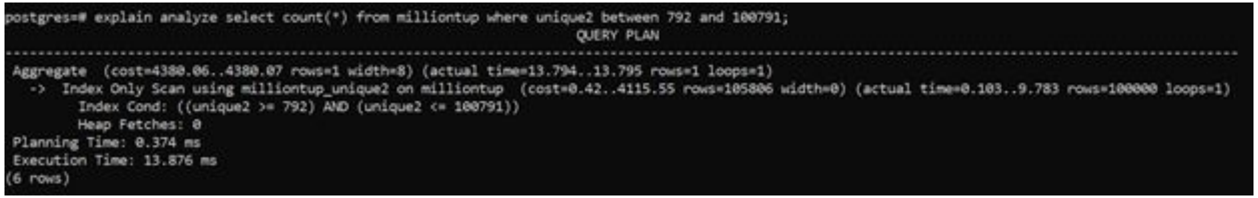
```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 10791;
                                                                     QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=454.90..454.91 rows=1 width=8) (actual time=3.255..3.256 rows=1 loops=1)
   ->  Index Only Scan using milliontup_unique2 on milliontup  (cost=0.42..427.63 rows=10910 width=0) (actual time=0.055..2.324 rows=10000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 10791))
         Heap Fetches: 0
 Planning Time: 3.103 ms
 Execution Time: 3.417 ms
(6 rows)


postgres=#
```

5%



```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 50792;
                                                                     QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2198.21..2198.22 rows=1 width=8) (actual time=8.325..8.326 rows=1 loops=1)
   ->  Index Only Scan using milliontup_unique2 on milliontup  (cost=0.42..2065.57 rows=53057 width=0) (actual time=0.036..5.924 rows=50001 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 50792))
         Heap Fetches: 0
 Planning Time: 0.305 ms
 Execution Time: 8.366 ms
(6 rows)
```

10%

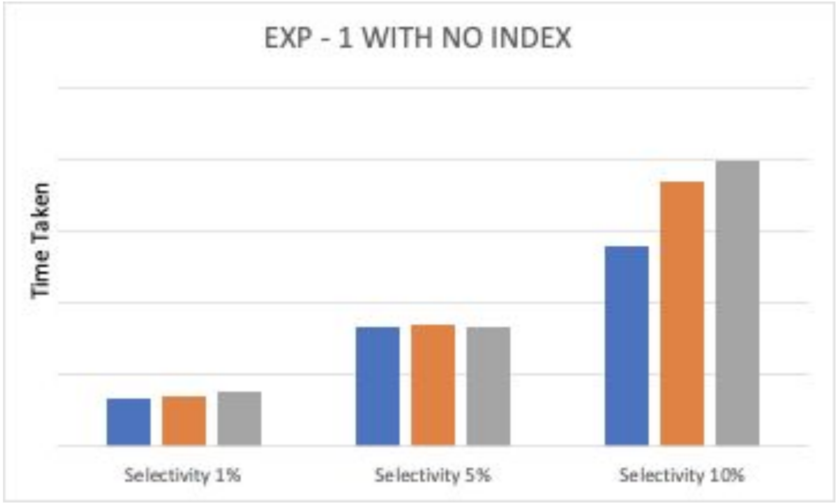


```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 100791;
                                                                     QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4380.06..4380.07 rows=1 width=8) (actual time=13.794..13.795 rows=1 loops=1)
   ->  Index Only Scan using milliontup_unique2 on milliontup  (cost=0.42..4115.55 rows=105806 width=0) (actual time=0.103..9.783 rows=100000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 100791))
         Heap Fetches: 0
 Planning Time: 0.374 ms
 Execution Time: 13.876 ms
(6 rows)
```

Experiment -1 Testing the 10% rule of thumb

With No Index

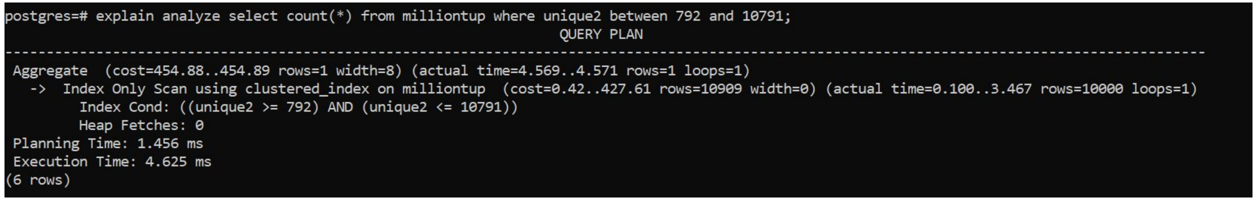
Selectivity	Execution Time for 1% in ms	Execution Time for 5% in ms	Execution Time for 10% in ms
1	3.417	8.366	13.876
2	3.493	8.444	12.908
3	4.101	8.228	18.420
4	3.256	8.373	21.251
5	3.814	8.724	19.859
Average	3.574	8.394	15.068



Experiment -1 Testing the 10% rule of thumb

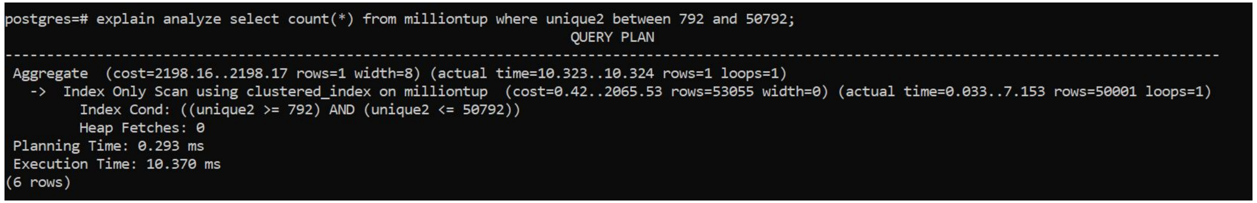
With clustered index

1%



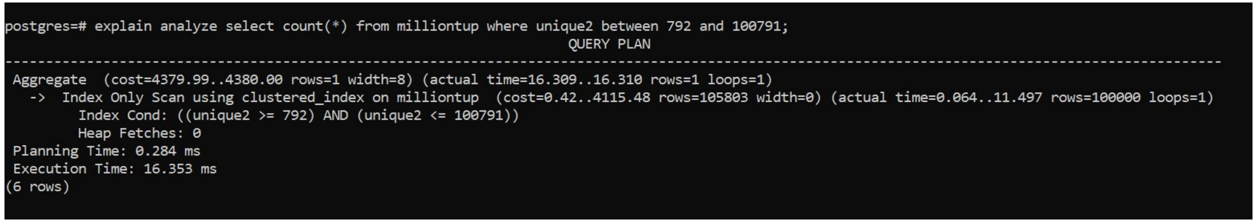
```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 10791;
                                                                QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=454.88..454.89 rows=1 width=8) (actual time=4.569..4.571 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..427.61 rows=10909 width=0) (actual time=0.100..3.467 rows=10000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 10791))
         Heap Fetches: 0
 Planning Time: 1.456 ms
 Execution Time: 4.625 ms
(6 rows)
```

5%



```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 50792;
                                                                QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2198.16..2198.17 rows=1 width=8) (actual time=10.323..10.324 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..2065.53 rows=53055 width=0) (actual time=0.033..7.153 rows=50001 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 50792))
         Heap Fetches: 0
 Planning Time: 0.293 ms
 Execution Time: 10.370 ms
(6 rows)
```

10%

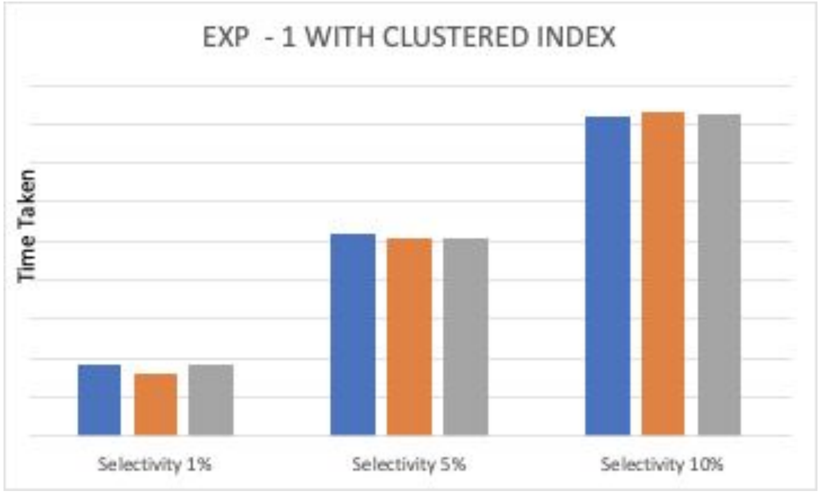


```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 100791;
                                                                QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4379.99..4380.00 rows=1 width=8) (actual time=16.309..16.310 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..4115.48 rows=105803 width=0) (actual time=0.064..11.497 rows=100000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 100791))
         Heap Fetches: 0
 Planning Time: 0.284 ms
 Execution Time: 16.353 ms
(6 rows)
```

Experiment -1 Testing the 10% rule of thumb

With clustered index

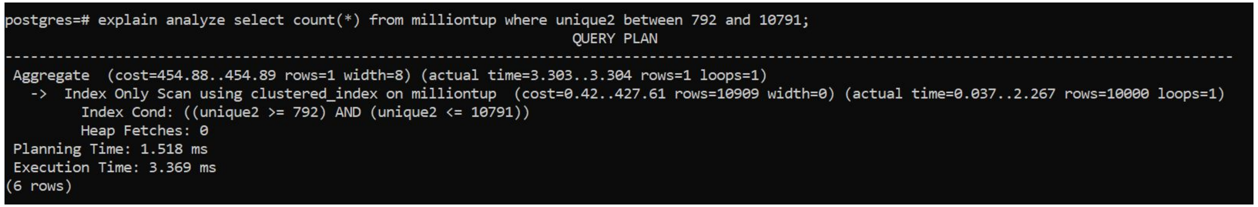
Selectivity	Execution Time for 1% in ms	Execution Time for 5% in ms	Execution Time for 10% in ms
1	4.625	10.370	16.353
2	3.091	9.135	16.598
3	3.651	10.121	17.513
4	3.260	10.120	16.529
5	3.642	11.601	15.792
Average	3.517	10.203	16.306



Experiment -1 Testing the 10% rule of thumb

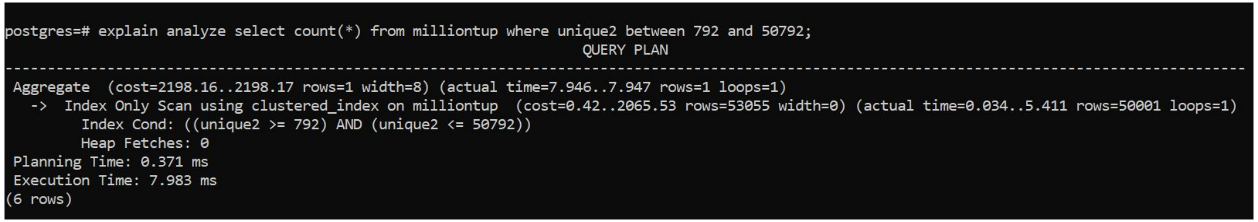
With unclustered index

1%



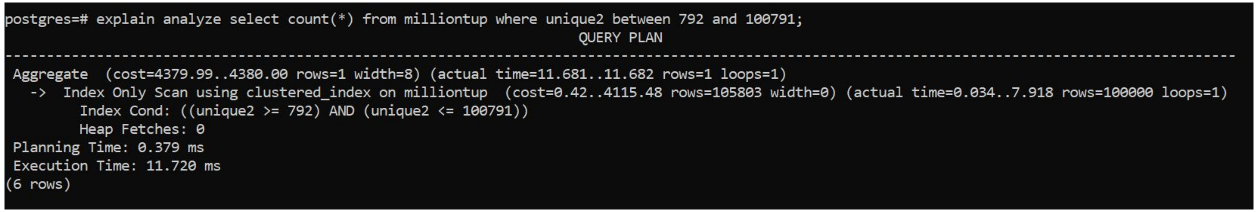
```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 10791;
                                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=454.88..454.89 rows=1 width=8) (actual time=3.303..3.304 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..427.61 rows=10909 width=0) (actual time=0.037..2.267 rows=10000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 10791))
         Heap Fetches: 0
 Planning Time: 1.518 ms
 Execution Time: 3.369 ms
(6 rows)
```

5%



```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 50792;
                                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2198.16..2198.17 rows=1 width=8) (actual time=7.946..7.947 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..2065.53 rows=53055 width=0) (actual time=0.034..5.411 rows=50001 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 50792))
         Heap Fetches: 0
 Planning Time: 0.371 ms
 Execution Time: 7.983 ms
(6 rows)
```

10%

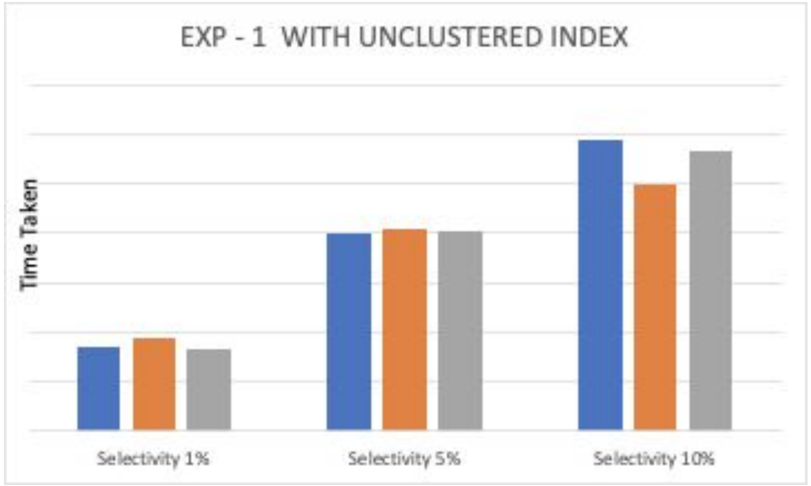


```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 100791;
                                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4379.99..4380.00 rows=1 width=8) (actual time=11.681..11.682 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..4115.48 rows=105803 width=0) (actual time=0.034..7.918 rows=100000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 100791))
         Heap Fetches: 0
 Planning Time: 0.379 ms
 Execution Time: 11.720 ms
(6 rows)
```

Experiment -1 Testing the 10% rule of thumb

With unclustered index

Selectivity	Execution Time for 1% in ms	Execution Time for 5% in ms	Execution Time for 10% in ms
1	3.369	7.983	11.720
2	3.791	8.143	9.687
3	4.356	7.566	12.242
4	3.279	8.452	9.988
5	3.016	8.042	11.280
Average	3.479	8.056	10.996



Experiment -2 Testing work_mem

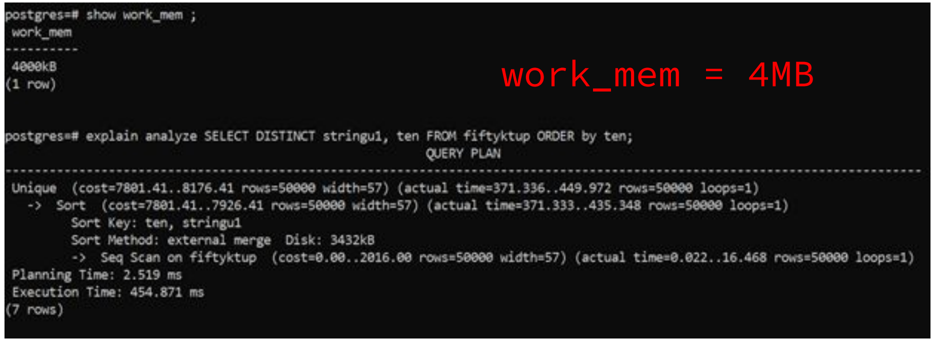
RELATION- **fiftyktup**

QUERY 1

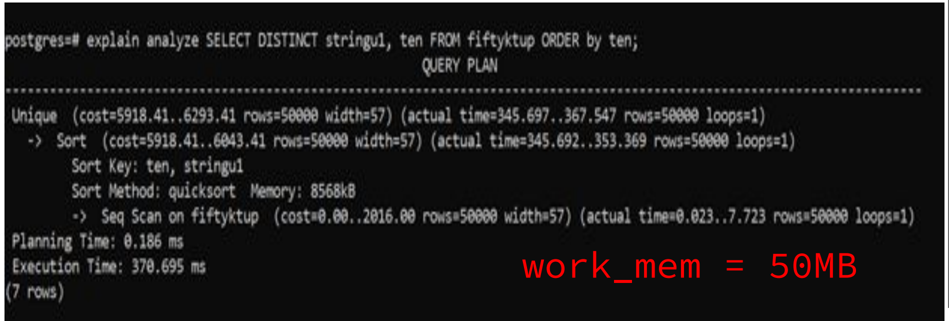
SELECT DISTINCT string1, ten

FROM fiftyktup

ORDER BY ten



```
postgres=# show work_mem ;
 work_mem
----------
 4000kB
(1 row)


postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM fiftyktup ORDER by ten;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Unique  (cost=7801.41..8176.41 rows=50000 width=57) (actual time=371.336..449.972 rows=50000 loops=1)
   ->  Sort  (cost=7801.41..7926.41 rows=50000 width=57) (actual time=371.333..435.348 rows=50000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: external merge  Disk: 3432kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.022..16.468 rows=50000 loops=1)
 Planning Time: 2.519 ms
 Execution Time: 454.871 ms
(7 rows)
```



```
postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM fiftyktup ORDER by ten;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Unique  (cost=5918.41..6293.41 rows=50000 width=57) (actual time=345.697..367.547 rows=50000 loops=1)
   ->  Sort  (cost=5918.41..6043.41 rows=50000 width=57) (actual time=345.692..353.369 rows=50000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: quicksort  Memory: 8568kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.023..7.723 rows=50000 loops=1)
 Planning Time: 0.186 ms
 Execution Time: 370.695 ms
(7 rows)
```

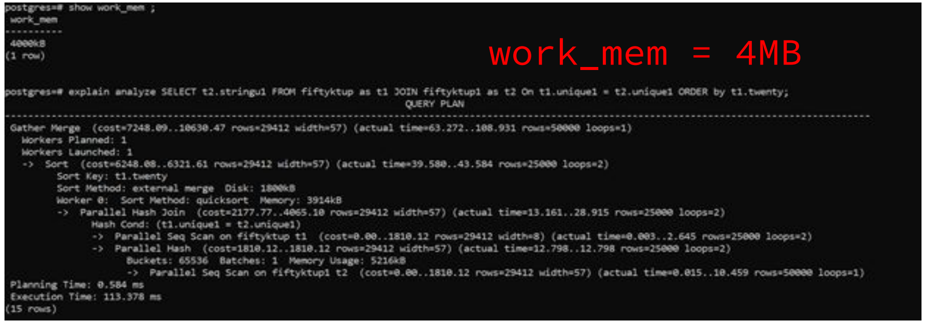
QUERY 2

SELECT t2.stringu1

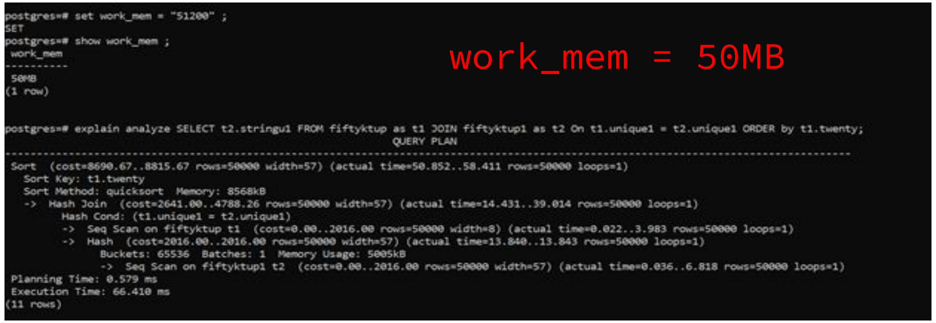
FROM fiftyktup **as** t1

JOIN fiftyktup1 **as** t2 **ON** t1.unique1 = t2.unique1

ORDER BY t1.twenty



```
postgres=# show work_mem ;
 work_mem
----------
 4000kB
(1 row)


postgres=# explain analyze SELECT t2.stringu1 FROM fiftyktup as t1 JOIN fiftyktup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Gather Merge  (cost=7248.09..10630.47 rows=29412 width=57) (actual time=63.272..108.931 rows=50000 loops=1)
   Workers Planned: 1
   Workers Launched: 1
   ->  Sort  (cost=6248.08..6321.61 rows=29412 width=57) (actual time=39.580..43.584 rows=25000 loops=2)
         Sort Key: t1.twenty
         Sort Method: external merge  Disk: 1800kB
         Worker 0:  Sort Method: quicksort  Memory: 3914kB
         ->  Parallel Hash Join  (cost=2177.77..4065.10 rows=29412 width=57) (actual time=13.161..28.915 rows=25000 loops=2)
               Hash Cond: (t1.unique1 = t2.unique1)
               ->  Parallel Seq Scan on fiftyktup t1  (cost=0.00..1810.12 rows=29412 width=8) (actual time=0.003..2.645 rows=25000 loops=2)
               ->  Parallel Hash  (cost=1810.12..1810.12 rows=29412 width=57) (actual time=12.798..12.798 rows=25000 loops=2)
                     Buckets: 65536  Batches: 1  Memory Usage: 5216kB
                     ->  Parallel Seq Scan on fiftyktup1 t2  (cost=0.00..1810.12 rows=29412 width=57) (actual time=0.015..10.459 rows=50000 loops=1)
 Planning Time: 0.584 ms
 Execution Time: 113.378 ms
(15 rows)
```



```
postgres=# set work_mem = "51200" ;
SET
postgres=# show work_mem ;
 work_mem
----------
 50MB
(1 row)


postgres=# explain analyze SELECT t2.stringu1 FROM fiftyktup as t1 JOIN fiftyktup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                                   QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=8690.67..8815.67 rows=50000 width=57) (actual time=50.852..58.411 rows=50000 loops=1)
   Sort Key: t1.twenty
   Sort Method: quicksort  Memory: 8568kB
   ->  Hash Join  (cost=2641.00..4788.26 rows=50000 width=57) (actual time=14.431..39.014 rows=50000 loops=1)
         Hash Cond: (t1.unique1 = t2.unique1)
         ->  Seq Scan on fiftyktup t1  (cost=0.00..2016.00 rows=50000 width=8) (actual time=0.022..3.983 rows=50000 loops=1)
         ->  Hash  (cost=2016.00..2016.00 rows=50000 width=57) (actual time=13.840..13.843 rows=50000 loops=1)
               Buckets: 65536  Batches: 1  Memory Usage: 5005kB
               ->  Seq Scan on fiftyktup1 t2  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.036..6.818 rows=50000 loops=1)
 Planning Time: 0.579 ms
 Execution Time: 66.410 ms
(11 rows)
```

Experiment -2 Testing work_mem

RELATION- **fiftyktup**

QUERY 1

SELECT DISTINCT string1, ten

FROM fiftyktup

ORDER BY ten

Query 1	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	454.871	370.695
2	447.982	382.481
3	450.396	410.963
4	466.862	394.628
5	450.262	402.742
Average	451.08	393.28

QUERY 2

SELECT t2.stringu1

FROM fiftyktup **as** t1

JOIN fiftyktup1 **as** t2 **ON** t1.unique1 = t2.unique1

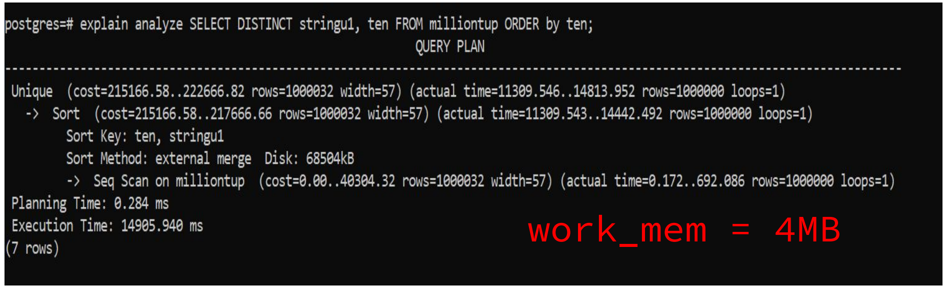
ORDER BY t1.twenty

Query 2	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	113.378	66.410
2	148.447	63.028
3	126.993	72.472
4	147.161	60.442
5	151.204	74.527
Average	140.87	67.30

Experiment -2 Testing work_mem

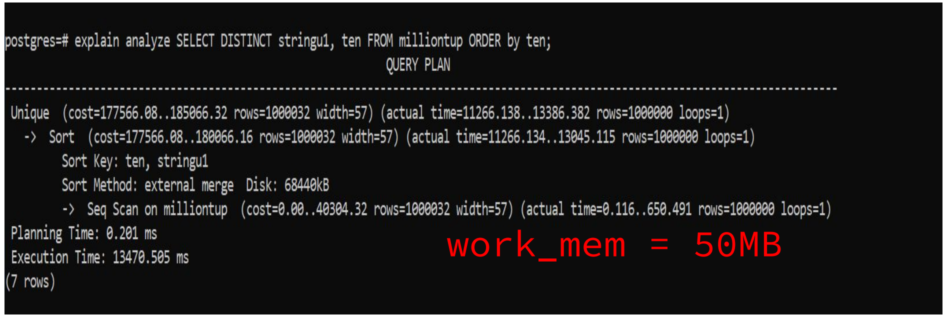
RELATION- **milliontup**

QUERY 1 **SELECT DISTINCT** string1, ten
FROM milliontup
ORDER BY ten



```
postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM milliontup ORDER by ten;
                                          QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Unique  (cost=215166.58..222666.82 rows=1000032 width=57) (actual time=11309.546..14813.952 rows=1000000 loops=1)
   ->  Sort  (cost=215166.58..217666.66 rows=1000032 width=57) (actual time=11309.543..14442.492 rows=1000000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: external merge  Disk: 68504kB
         ->  Seq Scan on milliontup  (cost=0.00..40304.32 rows=1000032 width=57) (actual time=0.172..692.086 rows=1000000 loops=1)
 Planning Time: 0.284 ms
 Execution Time: 14905.940 ms
(7 rows)
```

work_mem = 4MB

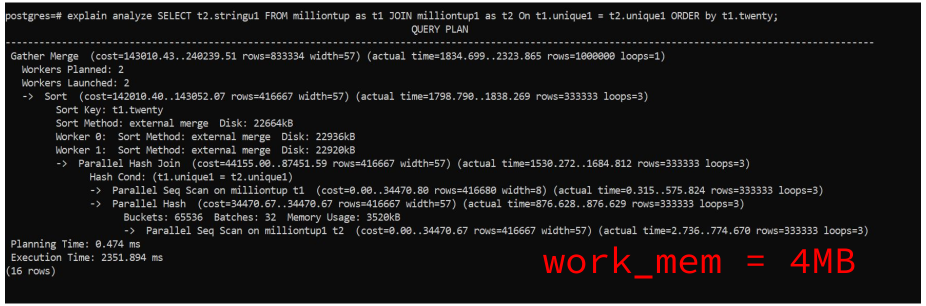


```
postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM milliontup ORDER by ten;
                                          QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Unique  (cost=177566.08..185066.32 rows=1000032 width=57) (actual time=11266.138..13386.382 rows=1000000 loops=1)
   ->  Sort  (cost=177566.08..180066.16 rows=1000032 width=57) (actual time=11266.134..13045.115 rows=1000000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: external merge  Disk: 68440kB
         ->  Seq Scan on milliontup  (cost=0.00..40304.32 rows=1000032 width=57) (actual time=0.116..650.491 rows=1000000 loops=1)
 Planning Time: 0.201 ms
 Execution Time: 13470.505 ms
(7 rows)
```

work_mem = 50MB

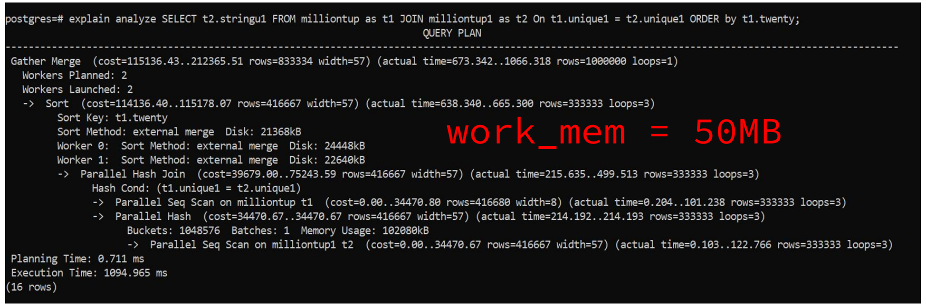
QUERY 2

SELECT t2.stringu1
FROM milliontup **AS** t1
JOIN milliontup **AS** t2 **ON** t1.unique1 = t2.unique1
ORDER by t1.twenty



```
postgres=# explain analyze SELECT t2.stringu1 FROM milliontup as t1 JOIN milliontup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Gather Merge  (cost=143010.43..240239.51 rows=833334 width=57) (actual time=1834.699..2323.865 rows=1000000 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Sort  (cost=142010.40..143052.07 rows=416667 width=57) (actual time=1798.790..1838.269 rows=333333 loops=3)
         Sort Key: t1.twenty
         Sort Method: external merge  Disk: 22664kB
         Worker 0:  Sort Method: external merge  Disk: 22936kB
         Worker 1:  Sort Method: external merge  Disk: 22920kB
         ->  Parallel Hash Join  (cost=44155.00..87451.59 rows=416667 width=57) (actual time=1530.272..1684.812 rows=333333 loops=3)
               Hash Cond: (t1.unique1 = t2.unique1)
               ->  Parallel Seq Scan on milliontup t1  (cost=0.00..34470.80 rows=416680 width=8) (actual time=0.315..575.824 rows=333333 loops=3)
               ->  Parallel Hash  (cost=34470.67..34470.67 rows=416667 width=57) (actual time=876.628..876.629 rows=333333 loops=3)
                     Buckets: 65536  Batches: 32  Memory Usage: 3520kB
                     ->  Parallel Seq Scan on milliontup1 t2  (cost=0.00..34470.67 rows=416667 width=57) (actual time=2.736..774.670 rows=333333 loops=3)
 Planning Time: 0.474 ms
 Execution Time: 2351.894 ms
(16 rows)
```

work_mem = 4MB



```
postgres=# explain analyze SELECT t2.stringu1 FROM milliontup as t1 JOIN milliontup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Gather Merge  (cost=115136.43..212365.51 rows=833334 width=57) (actual time=673.342..1066.318 rows=1000000 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Sort  (cost=114136.40..115178.07 rows=416667 width=57) (actual time=638.340..665.300 rows=333333 loops=3)
         Sort Key: t1.twenty
         Sort Method: external merge  Disk: 21368kB
         Worker 0:  Sort Method: external merge  Disk: 24448kB
         Worker 1:  Sort Method: external merge  Disk: 22640kB
         ->  Parallel Hash Join  (cost=39679.00..75243.59 rows=416667 width=57) (actual time=215.635..499.513 rows=333333 loops=3)
               Hash Cond: (t1.unique1 = t2.unique1)
               ->  Parallel Seq Scan on milliontup t1  (cost=0.00..34470.80 rows=416680 width=8) (actual time=0.204..101.238 rows=333333 loops=3)
               ->  Parallel Hash  (cost=34470.67..34470.67 rows=416667 width=57) (actual time=214.192..214.193 rows=333333 loops=3)
                     Buckets: 1048576  Batches: 1  Memory Usage: 102080kB
                     ->  Parallel Seq Scan on milliontup1 t2  (cost=0.00..34470.67 rows=416667 width=57) (actual time=0.103..122.766 rows=333333 loops=3)
 Planning Time: 0.711 ms
 Execution Time: 1094.965 ms
(16 rows)
```

work_mem = 50MB

Experiment -2 Testing work_mem

RELATION- **milliontup**

QUERY 1

SELECT DISTINCT string1, ten

FROM fiftyktup

ORDER BY ten

Query 1	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	14905.940	13470.505
2	14841.316	13283.937
3	14493.836	13169.354
4	14732.749	13673.226
5	14147.475	13103.774
Average	14689.3	13307.90

QUERY 2

SELECT t2.stringu1

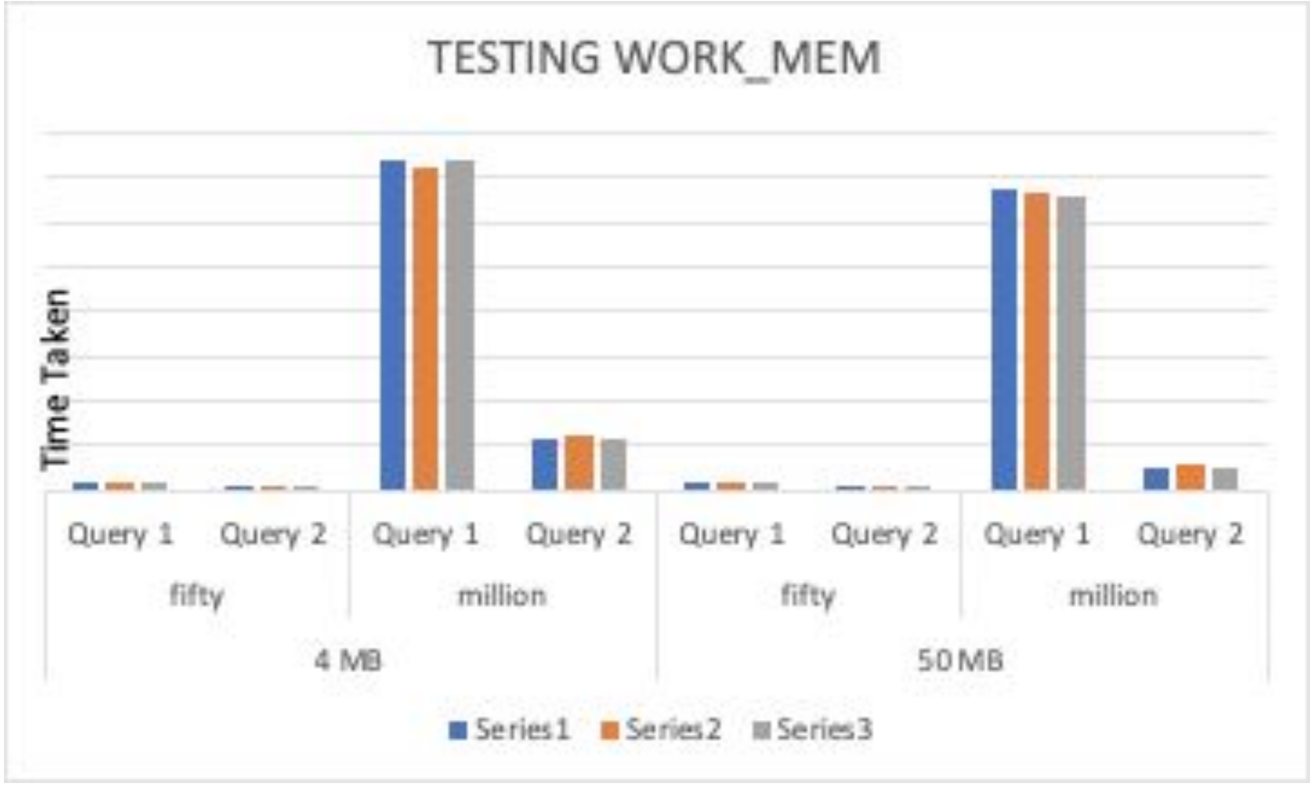
FROM milliontup **AS** t1

JOIN milliontup **AS** t2 **ON** t1.unique1 = t2.unique1

ORDER by t1.twenty

Query 2	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	2351.894	1094.965
2	2473.573	995.263
3	2146.563	1163.476
4	2573.284	958.947
5	2254.475	1003.465
Average	2359.98	1054.07

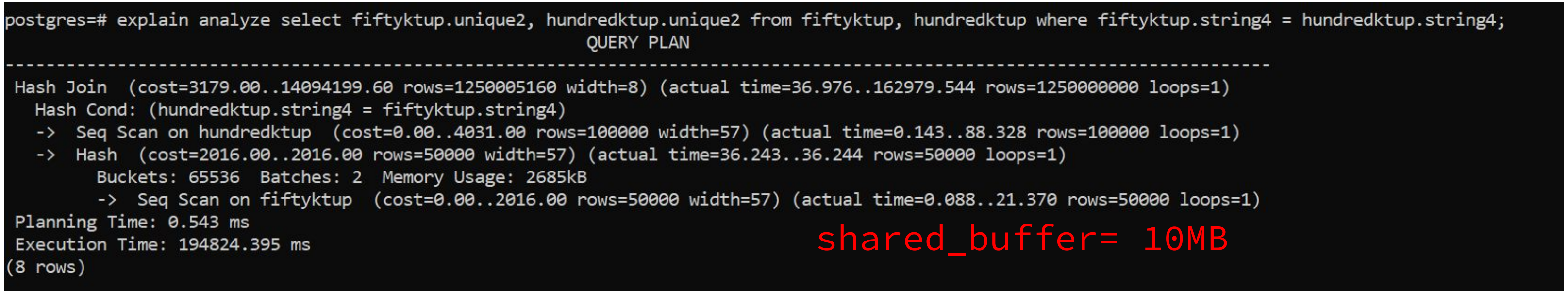
Experiment -2 Testing work_mem



Experiment -3 Testing shared_buffer

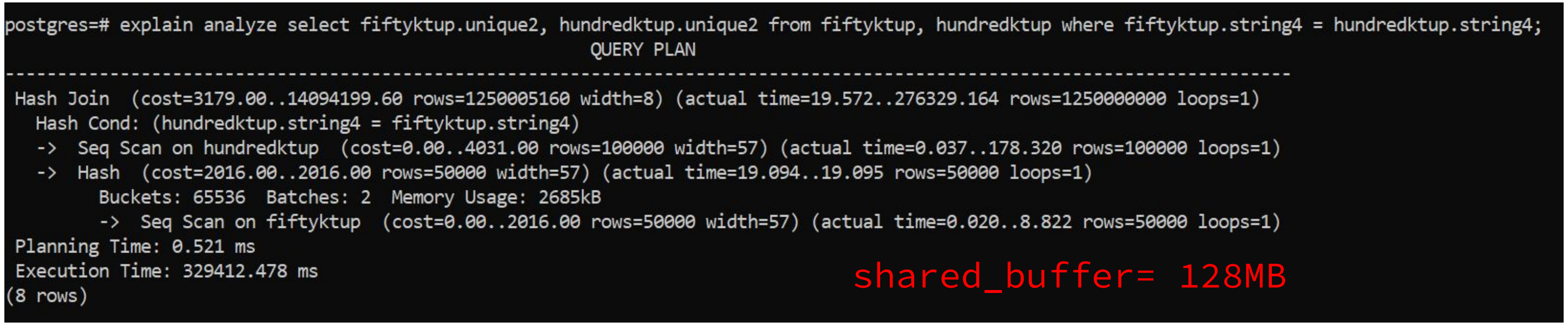
QUERY

SELECT fiftyktup.unique2, hundredktup.unique2
FROM fiftyktup, hundredktup
WHERE fiftyktup.string4 = hundredktup.string4



```
postgres=# explain analyze select fiftyktup.unique2, hundredktup.unique2 from fiftyktup, hundredktup where fiftyktup.string4 = hundredktup.string4;
                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=3179.00..14094199.60 rows=1250005160 width=8) (actual time=36.976..162979.544 rows=1250000000 loops=1)
   Hash Cond: (hundredktup.string4 = fiftyktup.string4)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=57) (actual time=0.143..88.328 rows=100000 loops=1)
   ->  Hash  (cost=2016.00..2016.00 rows=50000 width=57) (actual time=36.243..36.244 rows=50000 loops=1)
         Buckets: 65536  Batches: 2  Memory Usage: 2685kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.088..21.370 rows=50000 loops=1)
 Planning Time: 0.543 ms
 Execution Time: 194824.395 ms
(8 rows)
```

shared_buffer= 10MB

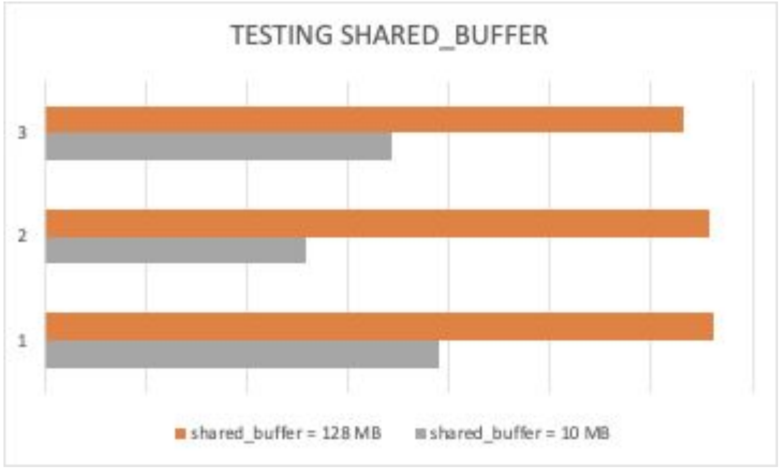


```
postgres=# explain analyze select fiftyktup.unique2, hundredktup.unique2 from fiftyktup, hundredktup where fiftyktup.string4 = hundredktup.string4;
                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=3179.00..14094199.60 rows=1250005160 width=8) (actual time=19.572..276329.164 rows=1250000000 loops=1)
   Hash Cond: (hundredktup.string4 = fiftyktup.string4)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=57) (actual time=0.037..178.320 rows=100000 loops=1)
   ->  Hash  (cost=2016.00..2016.00 rows=50000 width=57) (actual time=19.094..19.095 rows=50000 loops=1)
         Buckets: 65536  Batches: 2  Memory Usage: 2685kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.020..8.822 rows=50000 loops=1)
 Planning Time: 0.521 ms
 Execution Time: 329412.478 ms
(8 rows)
```

shared_buffer= 128MB

Experiment -3 Testing shared_buffer

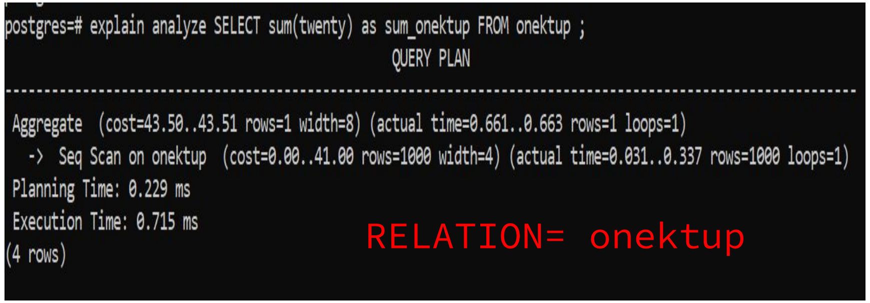
Query	shared_buffer = 10 MB	shared_buffer = 128 MB
1	194824.395	329412.478
2	129569.055	330479.180
3	266172.547	328394.253
4	171363.130	210198.582
5	136170.793	315924.851
Average	167452.773	324577.194



Experiment -4 Testing sizeup

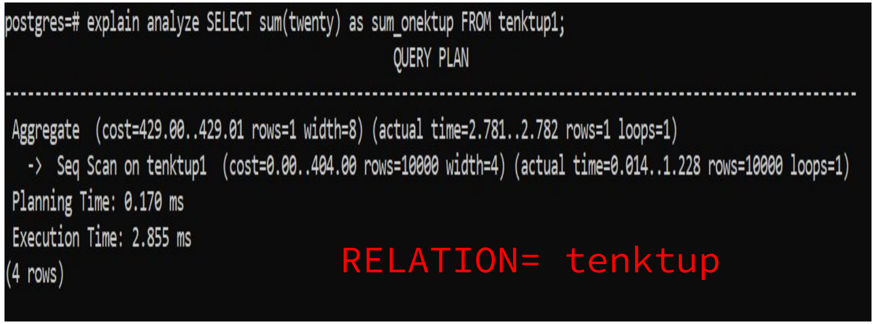
QUERY 1 **SELECT SUM** (twenty) as sum_twenty

FROM onektup /tenktup / fiftyktup /hundredktup/ milliontup



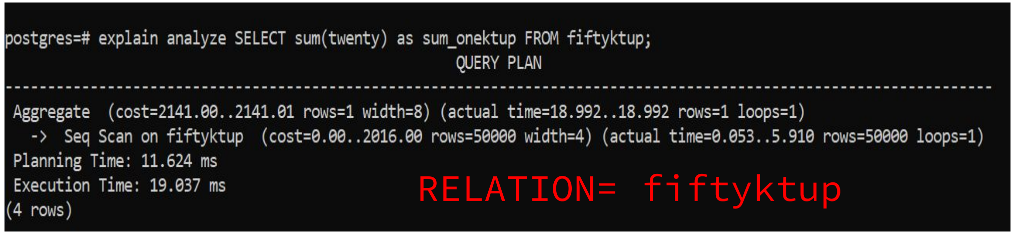
```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM onektup ;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=43.50..43.51 rows=1 width=8) (actual time=0.661..0.663 rows=1 loops=1)
   ->  Seq Scan on onektup  (cost=0.00..41.00 rows=1000 width=4) (actual time=0.031..0.337 rows=1000 loops=1)
 Planning Time: 0.229 ms
 Execution Time: 0.715 ms
(4 rows)
```

RELATION= onektup



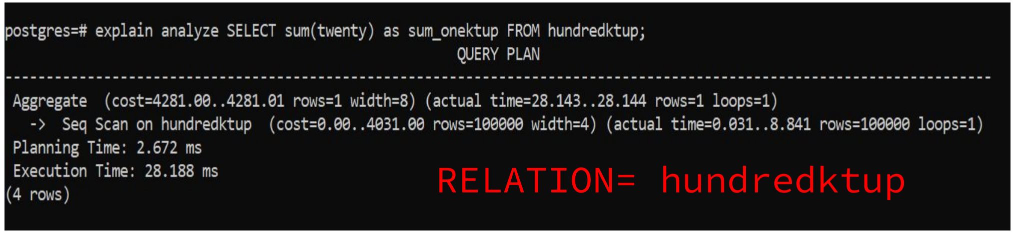
```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM tenktup1;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=429.00..429.01 rows=1 width=8) (actual time=2.781..2.782 rows=1 loops=1)
   ->  Seq Scan on tenktup1  (cost=0.00..404.00 rows=10000 width=4) (actual time=0.014..1.228 rows=10000 loops=1)
 Planning Time: 0.170 ms
 Execution Time: 2.855 ms
(4 rows)
```

RELATION= tenktup



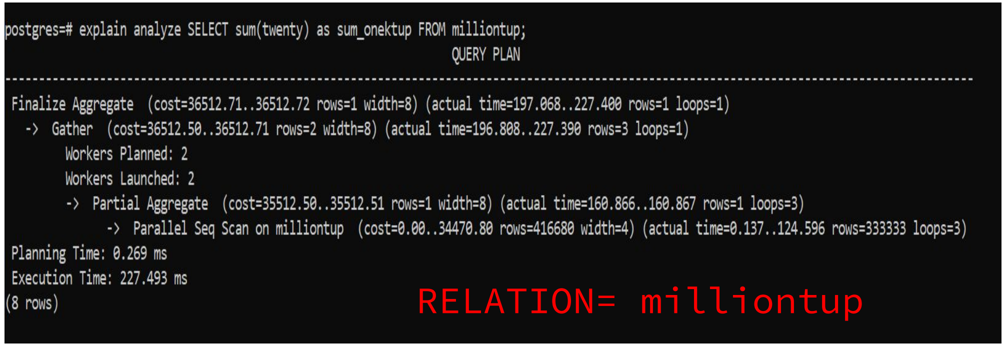
```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM fiftyktup;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2141.00..2141.01 rows=1 width=8) (actual time=18.992..18.992 rows=1 loops=1)
   ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=4) (actual time=0.053..5.910 rows=50000 loops=1)
 Planning Time: 11.624 ms
 Execution Time: 19.037 ms
(4 rows)
```

RELATION= fiftyktup



```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM hundredktup;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4281.00..4281.01 rows=1 width=8) (actual time=28.143..28.144 rows=1 loops=1)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=4) (actual time=0.031..8.841 rows=100000 loops=1)
 Planning Time: 2.672 ms
 Execution Time: 28.188 ms
(4 rows)
```

RELATION= hundredktup



```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM milliontup;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=36512.71..36512.72 rows=1 width=8) (actual time=197.068..227.400 rows=1 loops=1)
   ->  Gather  (cost=36512.50..36512.71 rows=2 width=8) (actual time=196.808..227.390 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=35512.50..35512.51 rows=1 width=8) (actual time=160.866..160.867 rows=1 loops=3)
               ->  Parallel Seq Scan on milliontup  (cost=0.00..34470.80 rows=416680 width=4) (actual time=0.137..124.596 rows=333333 loops=3)
 Planning Time: 0.269 ms
 Execution Time: 227.493 ms
(8 rows)
```

RELATION= milliontup

Experiment -4 Testing sizeup

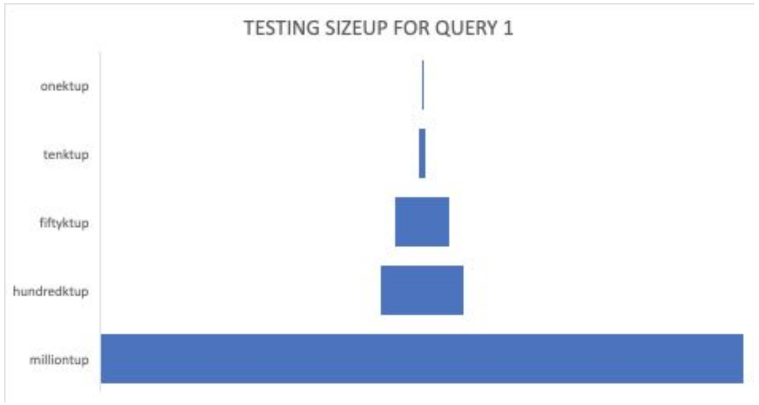
QUERY 1

SELECT SUM (twenty) as sum_twenty

FROM onektup /tenktup / fiftyktup

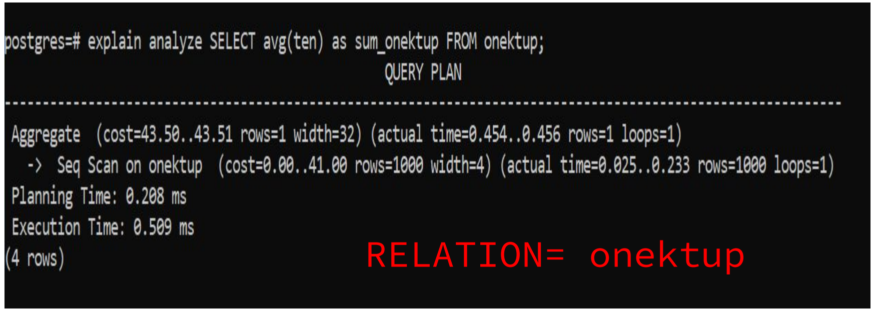
/hundredktup/ milliontup

Relation	onektup Execution Time (ms)	tenktup Execution Time (ms)	fiftyktup Execution Time (ms)	hundredktup Execution Time (ms)	milliontup Execution Time (ms)
1	0.715	2.855	19.037	28.188	227.493
2	0.383	2.991	19.558	34.718	216.907
3	0.377	3.355	14.877	33.271	222.260
4	0.381	2.809	16.104	30.225	218.913
5	0.360	2.557	14.137	27.130	216.133
Average	0.380	2.885	16.67	30.561	219.36



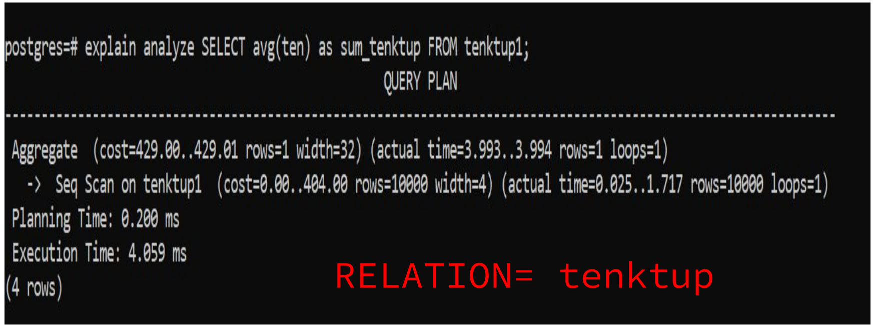
Experiment -4 Testing sizeup

QUERY 2 **SELECT AVG** (ten) as avg_ten
FROM onektup /tenktup / fiftyktup
/hundredktup/ milliontup



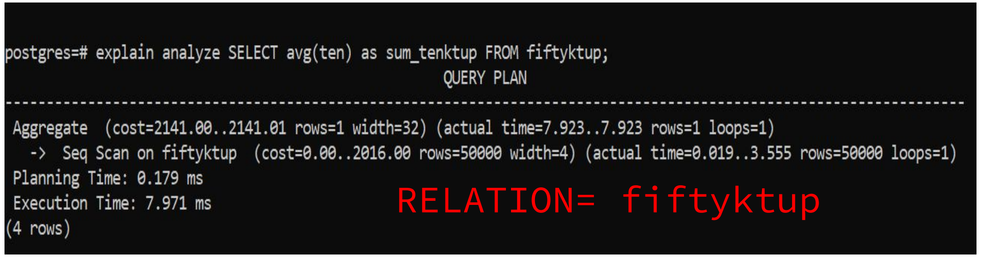
```
postgres=# explain analyze SELECT avg(ten) as sum_onektup FROM onektup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=43.50..43.51 rows=1 width=32) (actual time=0.454..0.456 rows=1 loops=1)
   ->  Seq Scan on onektup  (cost=0.00..41.00 rows=1000 width=4) (actual time=0.025..0.233 rows=1000 loops=1)
 Planning Time: 0.208 ms
 Execution Time: 0.509 ms
(4 rows)
```

RELATION= onektup



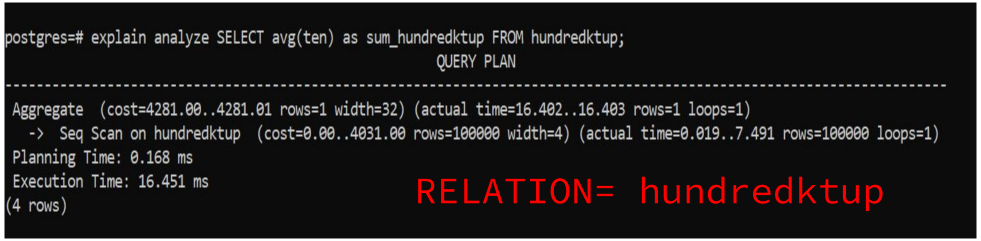
```
postgres=# explain analyze SELECT avg(ten) as sum_tenktup FROM tenktup1;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=429.00..429.01 rows=1 width=32) (actual time=3.993..3.994 rows=1 loops=1)
   ->  Seq Scan on tenktup1  (cost=0.00..404.00 rows=10000 width=4) (actual time=0.025..1.717 rows=10000 loops=1)
 Planning Time: 0.200 ms
 Execution Time: 4.059 ms
(4 rows)
```

RELATION= tenktup



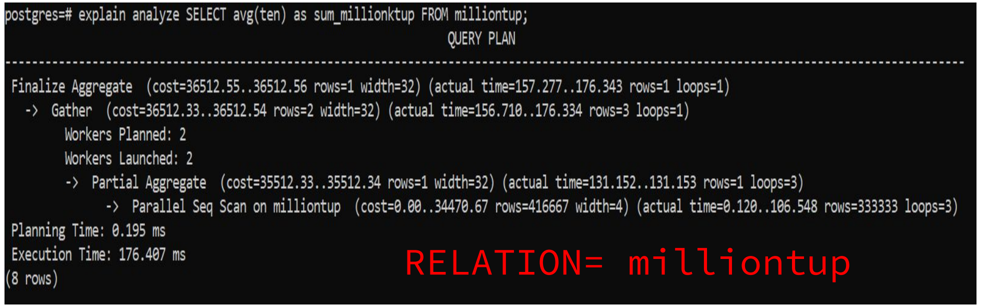
```
postgres=# explain analyze SELECT avg(ten) as sum_tenktup FROM fiftyktup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2141.00..2141.01 rows=1 width=32) (actual time=7.923..7.923 rows=1 loops=1)
   ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=4) (actual time=0.019..3.555 rows=50000 loops=1)
 Planning Time: 0.179 ms
 Execution Time: 7.971 ms
(4 rows)
```

RELATION= fiftyktup



```
postgres=# explain analyze SELECT avg(ten) as sum_hundredktup FROM hundredktup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4281.00..4281.01 rows=1 width=32) (actual time=16.402..16.403 rows=1 loops=1)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=4) (actual time=0.019..7.491 rows=100000 loops=1)
 Planning Time: 0.168 ms
 Execution Time: 16.451 ms
(4 rows)
```

RELATION= hundredktup



```
postgres=# explain analyze SELECT avg(ten) as sum_milliontup FROM milliontup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=36512.55..36512.56 rows=1 width=32) (actual time=157.277..176.343 rows=1 loops=1)
   ->  Gather  (cost=36512.33..36512.54 rows=2 width=32) (actual time=156.710..176.334 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=35512.33..35512.34 rows=1 width=32) (actual time=131.152..131.153 rows=1 loops=3)
               ->  Parallel Seq Scan on milliontup  (cost=0.00..34470.67 rows=416667 width=4) (actual time=0.120..106.548 rows=333333 loops=3)
 Planning Time: 0.195 ms
 Execution Time: 176.407 ms
(8 rows)
```

RELATION= milliontup

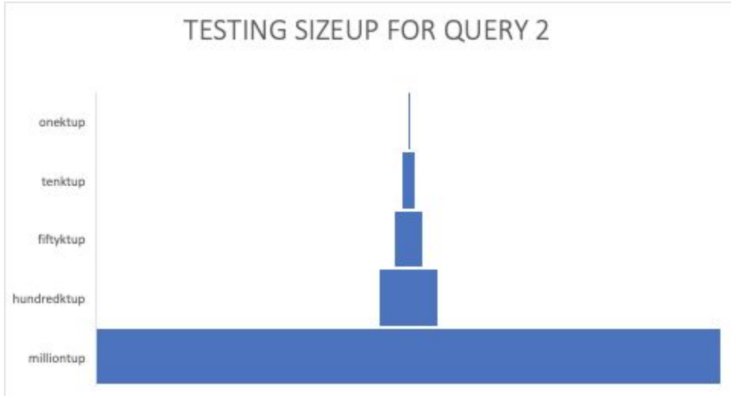
Experiment -4 Testing sizeup

QUERY 2 **SELECT AVG** (ten) as avg_ten

FROM onektup /tenktup / fiftyktup

/hundredktup/ milliontup

Relation	onektup Execution Time (ms)	tenktup Execution Time (ms)	fiftyktup Execution Time (ms)	hundredktup Execution Time (ms)	milliontup Execution Time (ms)
1	0.509	4.059	7.971	16.451	176.407
2	0.693	4.048	8.828	13.965	185.505
3	0.460	4.051	9.313	13.224	175.801
4	0.578	4.546	8.745	16.442	178.247
5	0.548	3.468	7.862	14.373	179.302
Average	0.55	4.052	8.52	14.926	177.78



Conclusions

- PostgreSQL selects the best query plan
- Selectivity is less than 10%, PostgreSQL does index scan
- Increase in work_mem, Decrease query execution time
- Increase in shared_buffer, Increase query execution time
- Increase in relation size, Increase query execution time

Lessons Learnt

- Wisconsin Benchmark
- Configuration Parameters / Memory Options
- Effects of database usage on execution of wide range of queries

Project Link:

Github Link

https://github.com/swet09/DB-Implementation-Project

Questions ?

APPENDIX

EXPERIMENTS

IN

Google Cloud Platform VM

Experiment -1 Testing the 10% rule of thumb

1% - select count(*) from milliontup where unique2 between 792 and 10791;

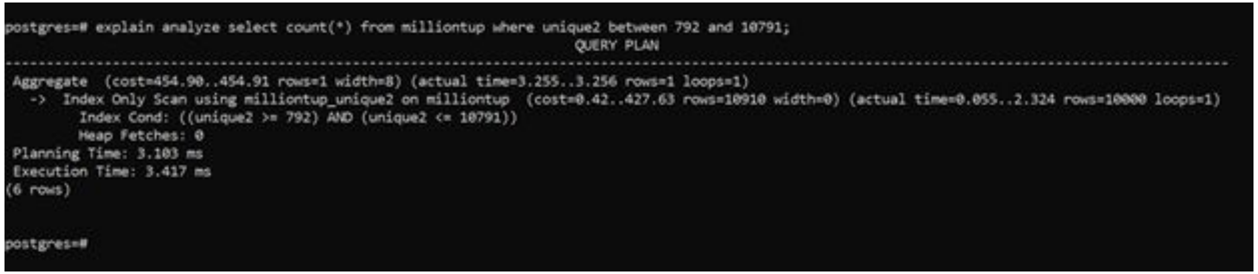
5% - select count(*) from milliontup where unique2 between 792 and 50792;

10% - explain analyze select count(*) from milliontup where unique2 between 792 and 100791;

Experiment -1 Testing the 10% rule of thumb

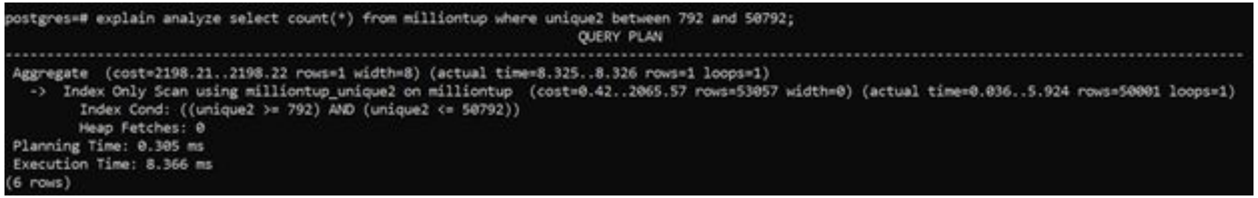
With No Index

1%



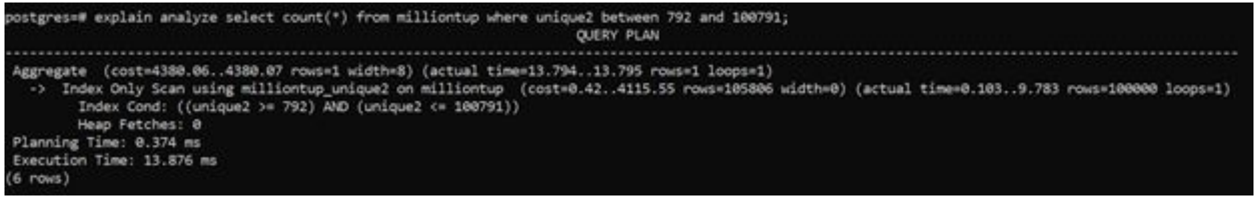
```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 10791;
                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=454.90..454.91 rows=1 width=8) (actual time=3.255..3.256 rows=1 loops=1)
   ->  Index Only Scan using milliontup_unique2 on milliontup  (cost=0.42..427.63 rows=10910 width=0) (actual time=0.055..2.324 rows=10000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 10791))
         Heap Fetches: 0
 Planning Time: 3.103 ms
 Execution Time: 3.417 ms
(6 rows)


postgres=#
```

5%



```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 50792;
                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2198.21..2198.22 rows=1 width=8) (actual time=8.325..8.326 rows=1 loops=1)
   ->  Index Only Scan using milliontup_unique2 on milliontup  (cost=0.42..2065.57 rows=53057 width=0) (actual time=0.036..5.924 rows=50001 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 50792))
         Heap Fetches: 0
 Planning Time: 0.305 ms
 Execution Time: 8.366 ms
(6 rows)
```

10%

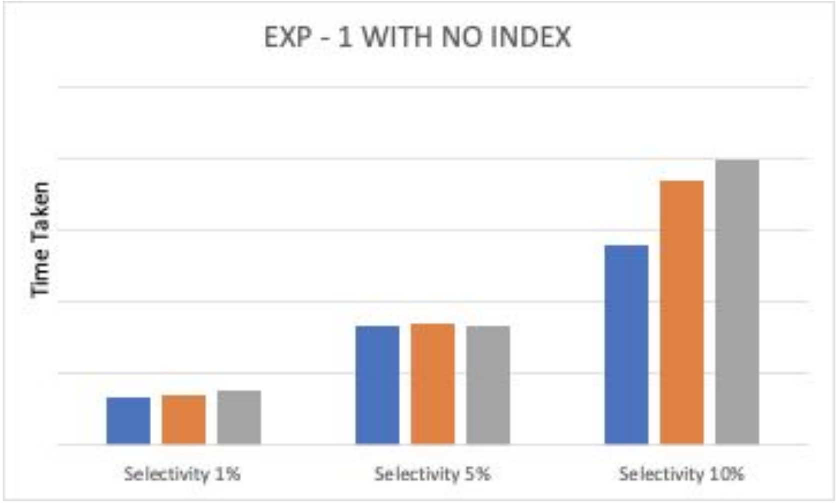


```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 100791;
                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4380.06..4380.07 rows=1 width=8) (actual time=13.794..13.795 rows=1 loops=1)
   ->  Index Only Scan using milliontup_unique2 on milliontup  (cost=0.42..4115.55 rows=105806 width=0) (actual time=0.103..9.783 rows=100000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 100791))
         Heap Fetches: 0
 Planning Time: 0.374 ms
 Execution Time: 13.876 ms
(6 rows)
```

Experiment -1 Testing the 10% rule of thumb

With No Index

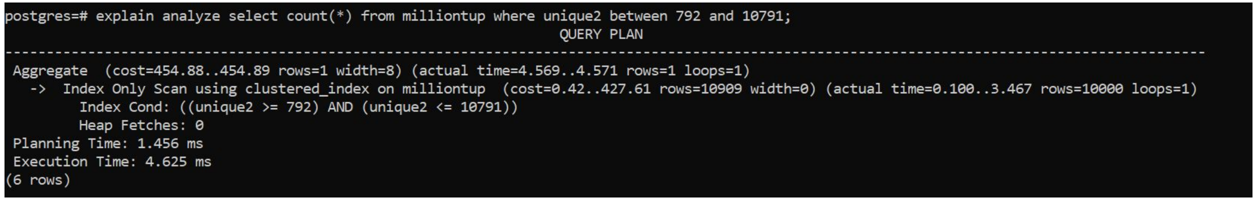
Selectivity	Execution Time for 1% in ms	Execution Time for 5% in ms	Execution Time for 10% in ms
1	3.417	8.366	13.876
2	3.493	8.444	12.908
3	4.101	8.228	18.420
4	3.256	8.373	21.251
5	3.814	8.724	19.859
Average	3.574	8.394	15.068



Experiment -1 Testing the 10% rule of thumb

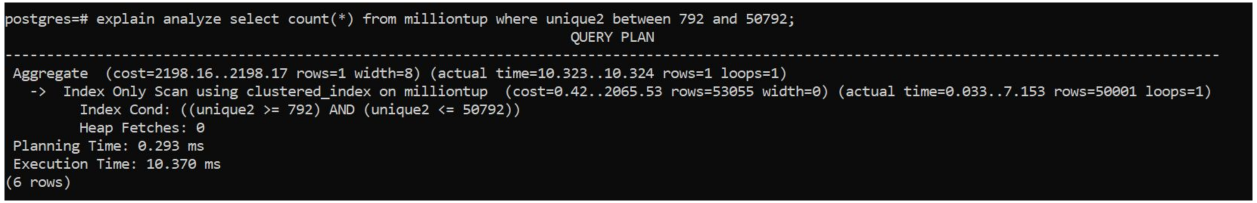
With clustered index

1%



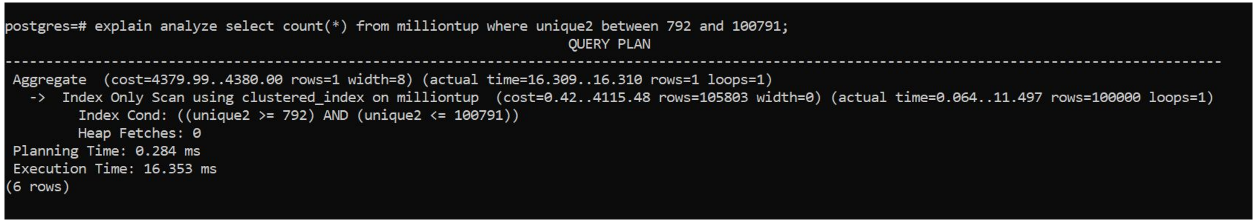
```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 10791;
                                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=454.88..454.89 rows=1 width=8) (actual time=4.569..4.571 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..427.61 rows=10909 width=0) (actual time=0.100..3.467 rows=10000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 10791))
         Heap Fetches: 0
 Planning Time: 1.456 ms
 Execution Time: 4.625 ms
(6 rows)
```

5%



```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 50792;
                                                                   QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2198.16..2198.17 rows=1 width=8) (actual time=10.323..10.324 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..2065.53 rows=53055 width=0) (actual time=0.033..7.153 rows=50001 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 50792))
         Heap Fetches: 0
 Planning Time: 0.293 ms
 Execution Time: 10.370 ms
(6 rows)
```

10%

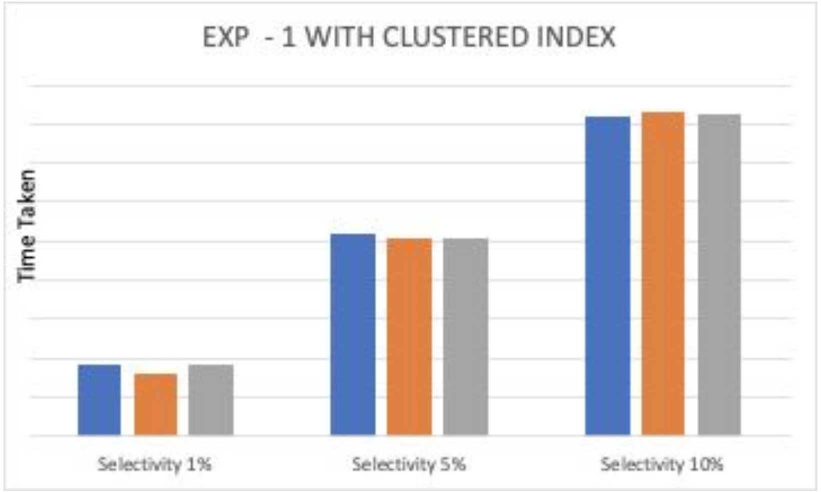


```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 100791;
                                                                   QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4379.99..4380.00 rows=1 width=8) (actual time=16.309..16.310 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..4115.48 rows=105803 width=0) (actual time=0.064..11.497 rows=100000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 100791))
         Heap Fetches: 0
 Planning Time: 0.284 ms
 Execution Time: 16.353 ms
(6 rows)
```

Experiment -1 Testing the 10% rule of thumb

With clustered index

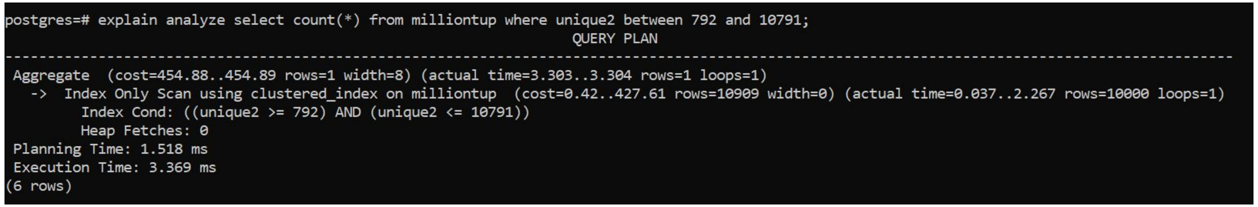
Selectivity	Execution Time for 1% in ms	Execution Time for 5% in ms	Execution Time for 10% in ms
1	4.625	10.370	16.353
2	3.091	9.135	16.598
3	3.651	10.121	17.513
4	3.260	10.120	16.529
5	3.642	11.601	15.792
Average	3.517	10.203	16.306



Experiment -1 Testing the 10% rule of thumb

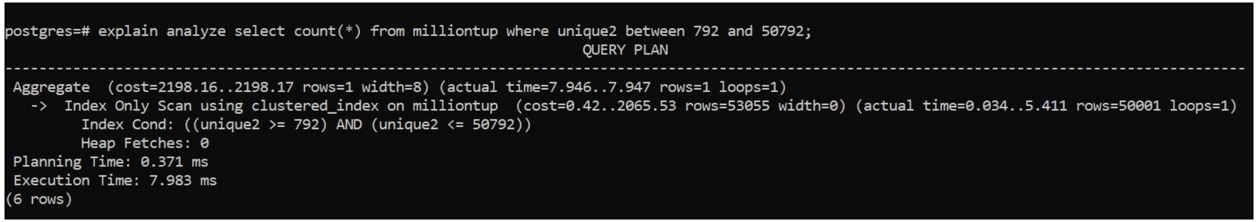
With unclustered index

1%



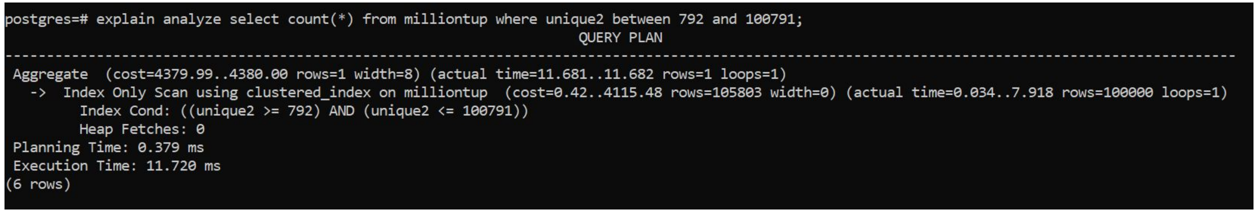
```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 10791;
                                                                      QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=454.88..454.89 rows=1 width=8) (actual time=3.303..3.304 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..427.61 rows=10909 width=0) (actual time=0.037..2.267 rows=10000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 10791))
         Heap Fetches: 0
 Planning Time: 1.518 ms
 Execution Time: 3.369 ms
(6 rows)
```

5%



```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 50792;
                                                                      QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2198.16..2198.17 rows=1 width=8) (actual time=7.946..7.947 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..2065.53 rows=53055 width=0) (actual time=0.034..5.411 rows=50001 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 50792))
         Heap Fetches: 0
 Planning Time: 0.371 ms
 Execution Time: 7.983 ms
(6 rows)
```

10%

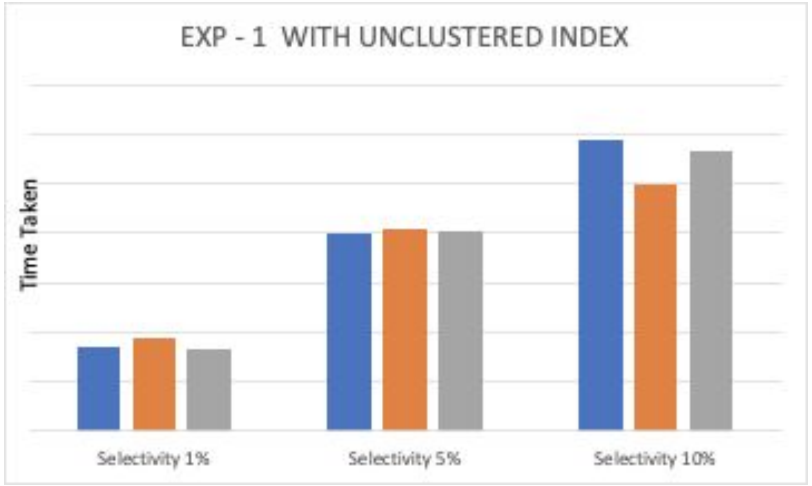


```
postgres=# explain analyze select count(*) from milliontup where unique2 between 792 and 100791;
                                                                      QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4379.99..4380.00 rows=1 width=8) (actual time=11.681..11.682 rows=1 loops=1)
   ->  Index Only Scan using clustered_index on milliontup  (cost=0.42..4115.48 rows=105803 width=0) (actual time=0.034..7.918 rows=100000 loops=1)
         Index Cond: ((unique2 >= 792) AND (unique2 <= 100791))
         Heap Fetches: 0
 Planning Time: 0.379 ms
 Execution Time: 11.720 ms
(6 rows)
```

Experiment -1 Testing the 10% rule of thumb

With unclustered index

Selectivity	Execution Time for 1% in ms	Executio n Time for 5% in ms	Execution Time for 10% in ms
1	3.369	7.983	11.720
2	3.791	8.143	9.687
3	4.356	7.566	12.242
4	3.279	8.452	9.988
5	3.016	8.042	11.280
Average	3.479	8.056	10.996



Experiment -2 Testing work_mem

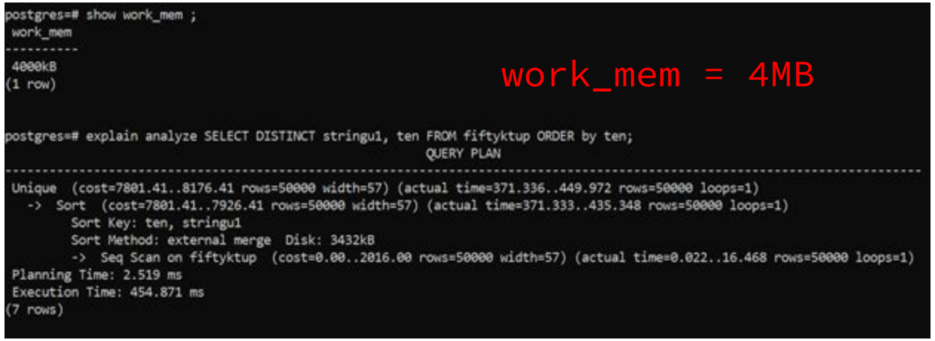
RELATION- **fiftyktup**

QUERY 1

SELECT DISTINCT string1, ten

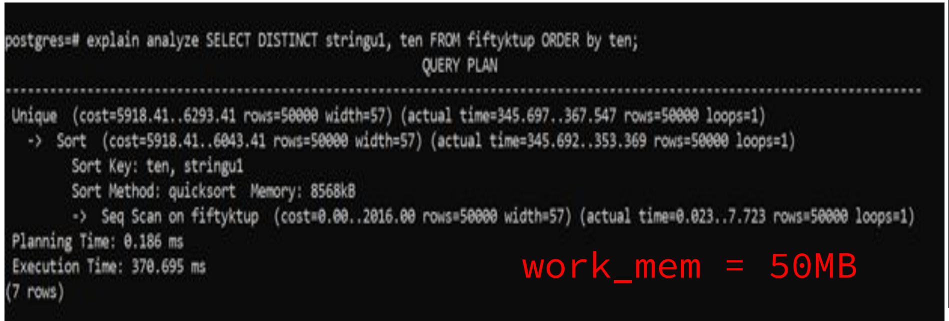
FROM fiftyktup

ORDER BY ten



```
postgres=# show work_mem ;
 work_mem
----------
 4000kB
(1 row)


postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM fiftyktup ORDER by ten;
                                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 Unique  (cost=7801.41..8176.41 rows=50000 width=57) (actual time=371.336..449.972 rows=50000 loops=1)
   ->  Sort  (cost=7801.41..7926.41 rows=50000 width=57) (actual time=371.333..435.348 rows=50000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: external merge  Disk: 3432kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.022..16.468 rows=50000 loops=1)
 Planning Time: 2.519 ms
 Execution Time: 454.871 ms
(7 rows)
```

work_mem = 4MB



```
postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM fiftyktup ORDER by ten;
                                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 Unique  (cost=5918.41..6293.41 rows=50000 width=57) (actual time=345.697..367.547 rows=50000 loops=1)
   ->  Sort  (cost=5918.41..6043.41 rows=50000 width=57) (actual time=345.692..353.369 rows=50000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: quicksort  Memory: 8568kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.023..7.723 rows=50000 loops=1)
 Planning Time: 0.186 ms
 Execution Time: 370.695 ms
(7 rows)
```

work_mem = 50MB

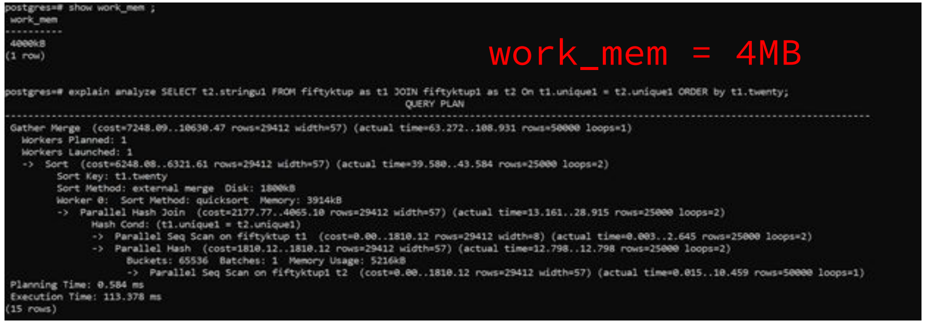
QUERY 2

SELECT t2.stringu1

FROM fiftyktup **as** t1

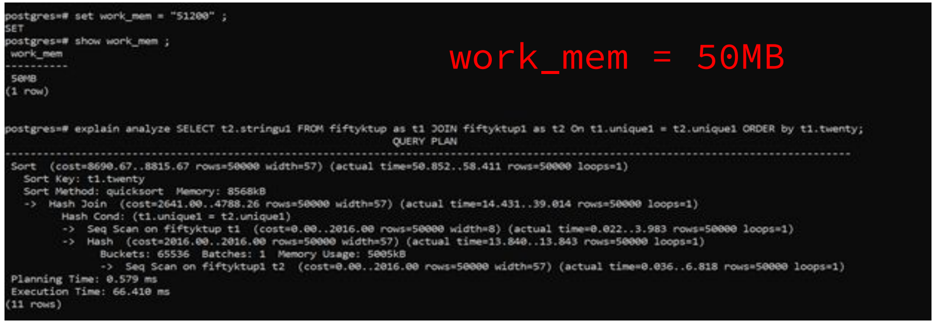
JOIN fiftyktup1 **as** t2 **ON** t1.unique1 = t2.unique1

ORDER BY t1.twenty



```
postgres=# show work_mem ;
 work_mem
----------
 4000kB
(1 row)


postgres=# explain analyze SELECT t2.stringu1 FROM fiftyktup as t1 JOIN fiftyktup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                                QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------
 Gather Merge  (cost=7248.09..10630.47 rows=29412 width=57) (actual time=63.272..108.931 rows=50000 loops=1)
   Workers Planned: 1
   Workers Launched: 1
   ->  Sort  (cost=6248.08..6321.61 rows=29412 width=57) (actual time=39.580..43.584 rows=25000 loops=2)
         Sort Key: t1.twenty
         Sort Method: external merge  Disk: 1800kB
         Worker 0:  Sort Method: quicksort  Memory: 3914kB
         ->  Parallel Hash Join  (cost=2177.77..4065.10 rows=29412 width=57) (actual time=13.161..28.915 rows=25000 loops=2)
               Hash Cond: (t1.unique1 = t2.unique1)
               ->  Parallel Seq Scan on fiftyktup t1  (cost=0.00..1810.12 rows=29412 width=8) (actual time=0.003..2.645 rows=25000 loops=2)
               ->  Parallel Hash  (cost=1810.12..1810.12 rows=29412 width=57) (actual time=12.798..12.798 rows=25000 loops=2)
                     Buckets: 65536  Batches: 1  Memory Usage: 5216kB
                     ->  Parallel Seq Scan on fiftyktup1 t2  (cost=0.00..1810.12 rows=29412 width=57) (actual time=0.015..10.459 rows=50000 loops=1)
 Planning Time: 0.584 ms
 Execution Time: 113.378 ms
(15 rows)
```

work_mem = 4MB



```
postgres=# set work_mem = "51200" ;
SET
postgres=# show work_mem ;
 work_mem
----------
 50MB
(1 row)


postgres=# explain analyze SELECT t2.stringu1 FROM fiftyktup as t1 JOIN fiftyktup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                               QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=8690.67..8815.67 rows=50000 width=57) (actual time=50.852..58.411 rows=50000 loops=1)
   Sort Key: t1.twenty
   Sort Method: quicksort  Memory: 8568kB
   ->  Hash Join  (cost=2641.00..4788.26 rows=50000 width=57) (actual time=14.431..39.014 rows=50000 loops=1)
         Hash Cond: (t1.unique1 = t2.unique1)
         ->  Seq Scan on fiftyktup t1  (cost=0.00..2016.00 rows=50000 width=8) (actual time=0.022..3.983 rows=50000 loops=1)
         ->  Hash  (cost=2016.00..2016.00 rows=50000 width=57) (actual time=13.840..13.843 rows=50000 loops=1)
               Buckets: 65536  Batches: 1  Memory Usage: 5005kB
               ->  Seq Scan on fiftyktup1 t2  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.036..6.818 rows=50000 loops=1)
 Planning Time: 0.579 ms
 Execution Time: 66.410 ms
(11 rows)
```

work_mem = 50MB

Experiment -2 Testing work_mem

RELATION- **fiftyktup**

QUERY 1

SELECT DISTINCT string1, ten
FROM fiftyktup
ORDER BY ten

Query 1	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	454.871	370.695
2	447.982	382.481
3	450.396	410.963
4	466.862	394.628
5	450.262	402.742
Average	451.08	393.28

QUERY 2

SELECT t2.stringu1
FROM fiftyktup **as** t1
JOIN fiftyktup1 **as** t2 **ON** t1.unique1 = t2.unique1
ORDER BY t1.twenty

Query 2	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	113.378	66.410
2	148.447	63.028
3	126.993	72.472
4	147.161	60.442
5	151.204	74.527
Average	140.87	67.30

Experiment -2 Testing work_mem

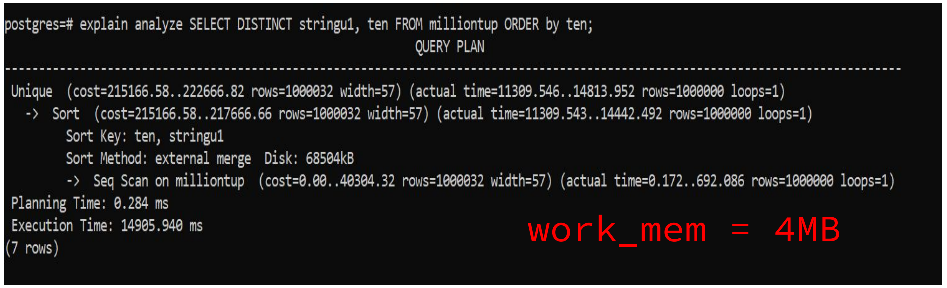
RELATION- **milliontup**

QUERY 1

SELECT DISTINCT string1, ten

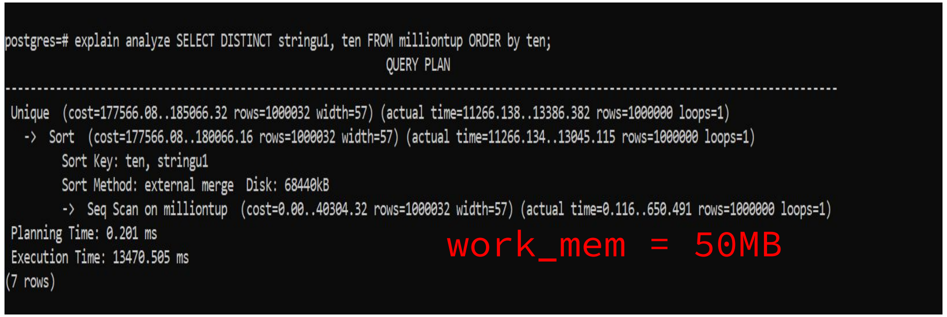
FROM milliontup

ORDER BY ten



```
postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM milliontup ORDER by ten;
                                                  QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Unique  (cost=215166.58..222666.82 rows=1000032 width=57) (actual time=11309.546..14813.952 rows=1000000 loops=1)
   ->  Sort  (cost=215166.58..217666.66 rows=1000032 width=57) (actual time=11309.543..14442.492 rows=1000000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: external merge  Disk: 68504kB
         ->  Seq Scan on milliontup  (cost=0.00..40304.32 rows=1000032 width=57) (actual time=0.172..692.086 rows=1000000 loops=1)
 Planning Time: 0.284 ms
 Execution Time: 14905.940 ms
(7 rows)
```

work_mem = 4MB



```
postgres=# explain analyze SELECT DISTINCT stringu1, ten FROM milliontup ORDER by ten;
                                                  QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Unique  (cost=177566.08..185066.32 rows=1000032 width=57) (actual time=11266.138..13386.382 rows=1000000 loops=1)
   ->  Sort  (cost=177566.08..180066.16 rows=1000032 width=57) (actual time=11266.134..13045.115 rows=1000000 loops=1)
         Sort Key: ten, stringu1
         Sort Method: external merge  Disk: 68440kB
         ->  Seq Scan on milliontup  (cost=0.00..40304.32 rows=1000032 width=57) (actual time=0.116..650.491 rows=1000000 loops=1)
 Planning Time: 0.201 ms
 Execution Time: 13470.505 ms
(7 rows)
```

work_mem = 50MB

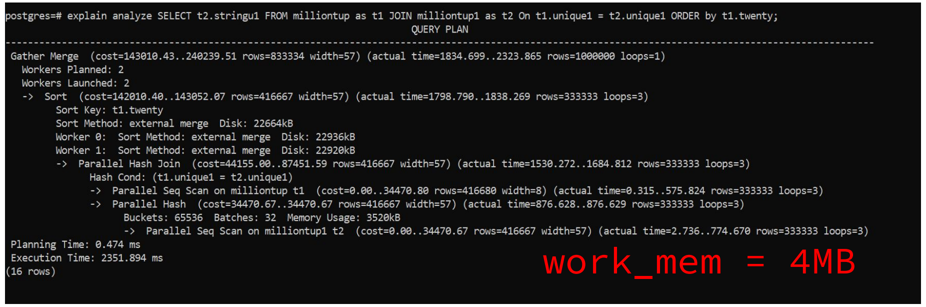
QUERY 2

SELECT t2.stringu1

FROM milliontup **AS** t1

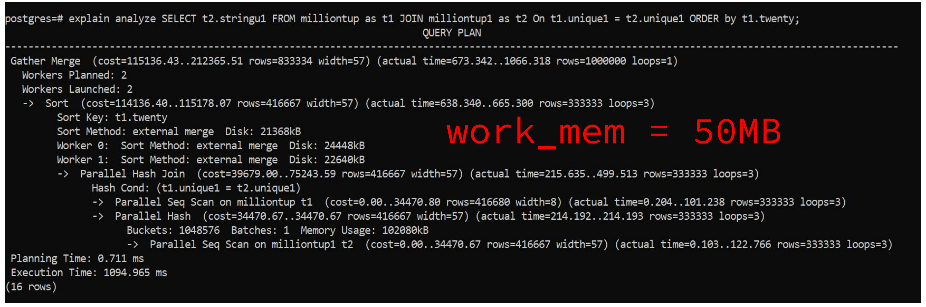
JOIN milliontup **AS** t2 **ON** t1.unique1 = t2.unique1

ORDER by t1.twenty



```
postgres=# explain analyze SELECT t2.stringu1 FROM milliontup as t1 JOIN milliontup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                  QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Gather Merge  (cost=143010.43..240239.51 rows=833334 width=57) (actual time=1834.699..2323.865 rows=1000000 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Sort  (cost=142010.40..143052.07 rows=416667 width=57) (actual time=1798.790..1838.269 rows=333333 loops=3)
         Sort Key: t1.twenty
         Sort Method: external merge  Disk: 22664kB
         Worker 0:  Sort Method: external merge  Disk: 22936kB
         Worker 1:  Sort Method: external merge  Disk: 22920kB
         ->  Parallel Hash Join  (cost=44155.00..87451.59 rows=416667 width=57) (actual time=1530.272..1684.812 rows=333333 loops=3)
               Hash Cond: (t1.unique1 = t2.unique1)
               ->  Parallel Seq Scan on milliontup t1  (cost=0.00..34470.80 rows=416680 width=8) (actual time=0.315..575.824 rows=333333 loops=3)
               ->  Parallel Hash  (cost=34470.67..34470.67 rows=416667 width=57) (actual time=876.628..876.629 rows=333333 loops=3)
                     Buckets: 65536  Batches: 32  Memory Usage: 3520kB
                     ->  Parallel Seq Scan on milliontup1 t2  (cost=0.00..34470.67 rows=416667 width=57) (actual time=2.736..774.670 rows=333333 loops=3)
 Planning Time: 0.474 ms
 Execution Time: 2351.894 ms
(16 rows)
```

work_mem = 4MB



```
postgres=# explain analyze SELECT t2.stringu1 FROM milliontup as t1 JOIN milliontup1 as t2 On t1.unique1 = t2.unique1 ORDER by t1.twenty;
                                                  QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Gather Merge  (cost=115136.43..212365.51 rows=833334 width=57) (actual time=673.342..1066.318 rows=1000000 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Sort  (cost=114136.40..115178.07 rows=416667 width=57) (actual time=638.340..665.300 rows=333333 loops=3)
         Sort Key: t1.twenty
         Sort Method: external merge  Disk: 21368kB
         Worker 0:  Sort Method: external merge  Disk: 24448kB
         Worker 1:  Sort Method: external merge  Disk: 22640kB
         ->  Parallel Hash Join  (cost=39679.00..75243.59 rows=416667 width=57) (actual time=215.635..499.513 rows=333333 loops=3)
               Hash Cond: (t1.unique1 = t2.unique1)
               ->  Parallel Seq Scan on milliontup t1  (cost=0.00..34470.80 rows=416680 width=8) (actual time=0.204..101.238 rows=333333 loops=3)
               ->  Parallel Hash  (cost=34470.67..34470.67 rows=416667 width=57) (actual time=214.192..214.193 rows=333333 loops=3)
                     Buckets: 1048576  Batches: 1  Memory Usage: 102080kB
                     ->  Parallel Seq Scan on milliontup1 t2  (cost=0.00..34470.67 rows=416667 width=57) (actual time=0.103..122.766 rows=333333 loops=3)
 Planning Time: 0.711 ms
 Execution Time: 1094.965 ms
(16 rows)
```

work_mem = 50MB

Experiment -2 Testing work_mem

RELATION- **milliontup**

QUERY 1

SELECT DISTINCT string1, ten

FROM fiftyktup

ORDER BY ten

Query 1	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	14905.940	13470.505
2	14841.316	13283.937
3	14493.836	13169.354
4	14732.749	13673.226
5	14147.475	13103.774
Average	14689.3	13307.90

QUERY 2

SELECT t2.stringu1

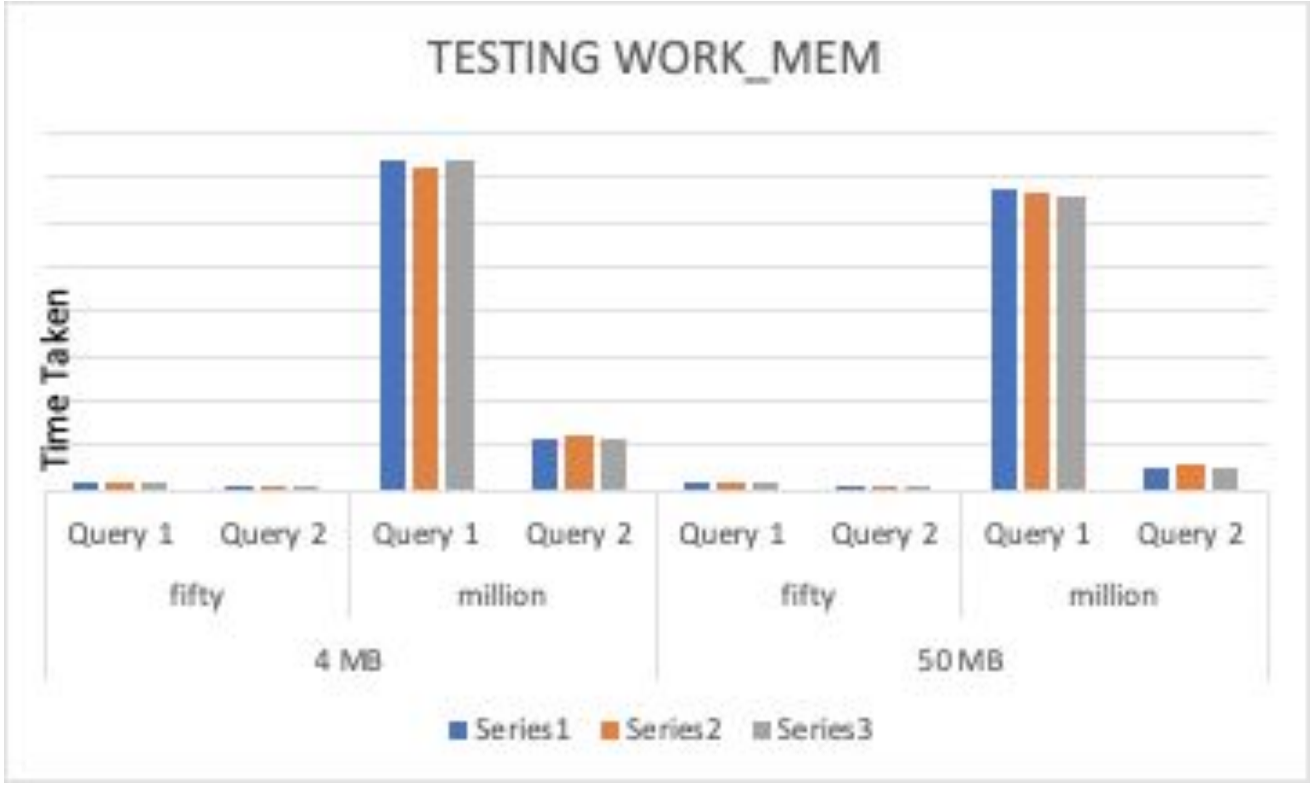
FROM milliontup **AS** t1

JOIN milliontup **AS** t2 **ON** t1.unique1 = t2.unique1

ORDER by t1.twenty

Query 2	**work_mem = 4MB** TIME (MS)	**work_mem = 50MB** TIME (MS)
1	2351.894	1094.965
2	2473.573	995.263
3	2146.563	1163.476
4	2573.284	958.947
5	2254.475	1003.465
Average	2359.98	1054.07

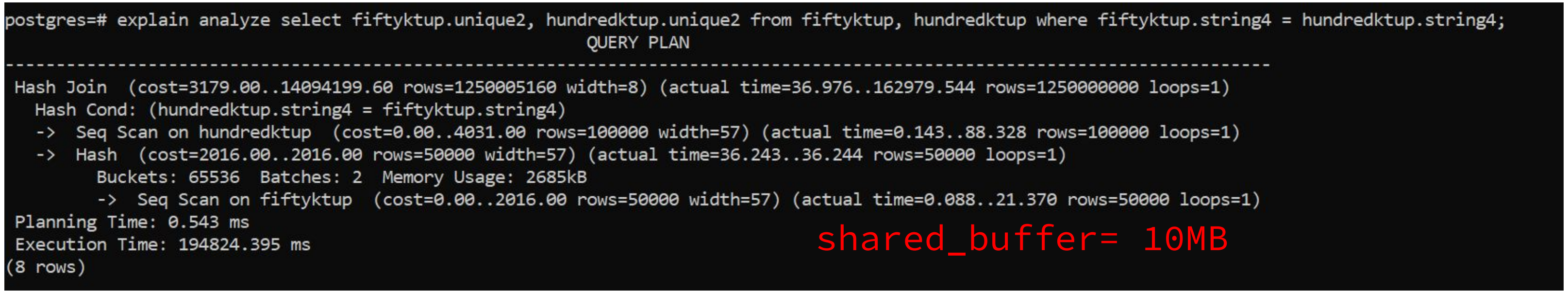
Experiment -2 Testing work_mem



Experiment -3 Testing shared_buffer

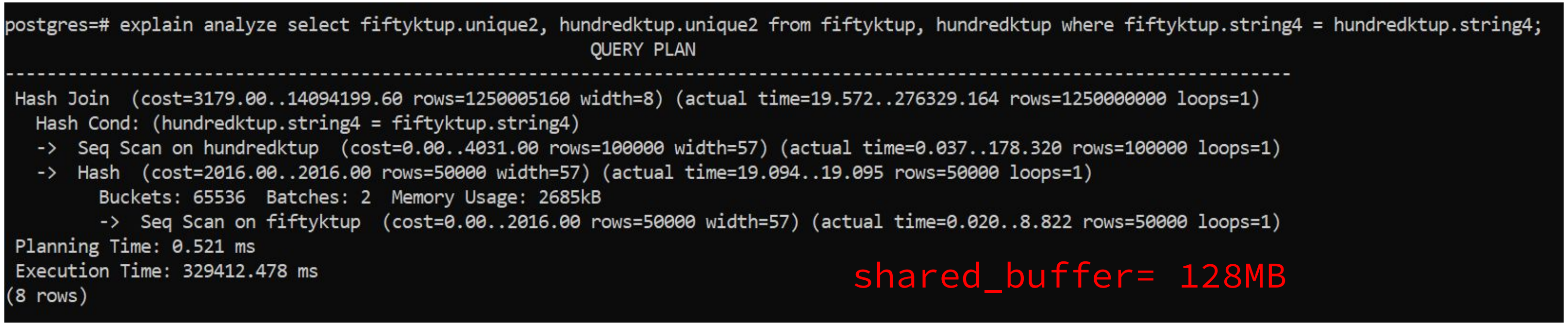
QUERY

SELECT fiftyktup.unique2, hundredktup.unique2
FROM fiftyktup, hundredktup
WHERE fiftyktup.string4 = hundredktup.string4



```
postgres=# explain analyze select fiftyktup.unique2, hundredktup.unique2 from fiftyktup, hundredktup where fiftyktup.string4 = hundredktup.string4;
                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=3179.00..14094199.60 rows=1250005160 width=8) (actual time=36.976..162979.544 rows=1250000000 loops=1)
   Hash Cond: (hundredktup.string4 = fiftyktup.string4)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=57) (actual time=0.143..88.328 rows=100000 loops=1)
   ->  Hash  (cost=2016.00..2016.00 rows=50000 width=57) (actual time=36.243..36.244 rows=50000 loops=1)
         Buckets: 65536  Batches: 2  Memory Usage: 2685kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.088..21.370 rows=50000 loops=1)
 Planning Time: 0.543 ms
 Execution Time: 194824.395 ms
(8 rows)
```

shared_buffer= 10MB

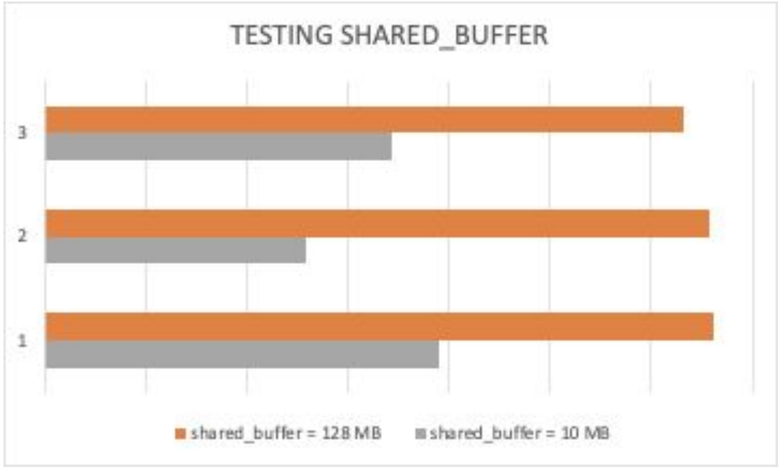


```
postgres=# explain analyze select fiftyktup.unique2, hundredktup.unique2 from fiftyktup, hundredktup where fiftyktup.string4 = hundredktup.string4;
                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=3179.00..14094199.60 rows=1250005160 width=8) (actual time=19.572..276329.164 rows=1250000000 loops=1)
   Hash Cond: (hundredktup.string4 = fiftyktup.string4)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=57) (actual time=0.037..178.320 rows=100000 loops=1)
   ->  Hash  (cost=2016.00..2016.00 rows=50000 width=57) (actual time=19.094..19.095 rows=50000 loops=1)
         Buckets: 65536  Batches: 2  Memory Usage: 2685kB
         ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=57) (actual time=0.020..8.822 rows=50000 loops=1)
 Planning Time: 0.521 ms
 Execution Time: 329412.478 ms
(8 rows)
```

shared_buffer= 128MB

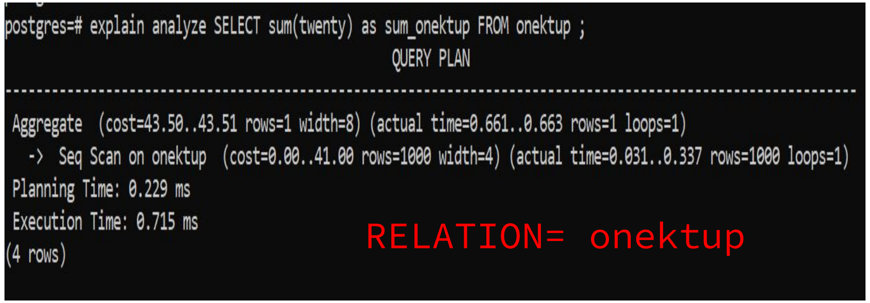
Experiment -3 Testing shared_buffer

Query	shared_buffer = 10 MB	shared_buffer = 128 MB
1	194824.395	329412.478
2	129569.055	330479.180
3	266172.547	328394.253
4	171363.130	210198.582
5	136170.793	315924.851
Average	167452.773	324577.194



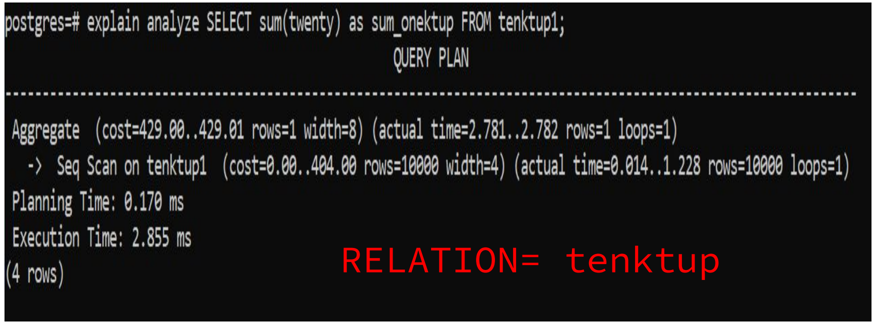
Experiment -4 Testing sizeup

QUERY 1 **SELECT SUM** (twenty) as sum_twenty
FROM onektup /tenktup / fiftyktup
/hundredktup/ milliontup



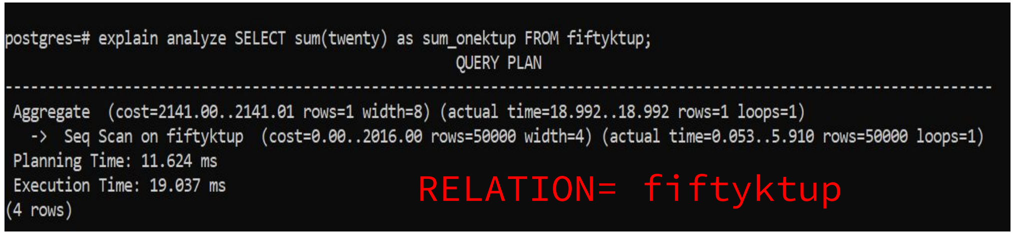
```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM onektup ;
                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=43.50..43.51 rows=1 width=8) (actual time=0.661..0.663 rows=1 loops=1)
   ->  Seq Scan on onektup  (cost=0.00..41.00 rows=1000 width=4) (actual time=0.031..0.337 rows=1000 loops=1)
 Planning Time: 0.229 ms
 Execution Time: 0.715 ms
(4 rows)
```

RELATION= onektup



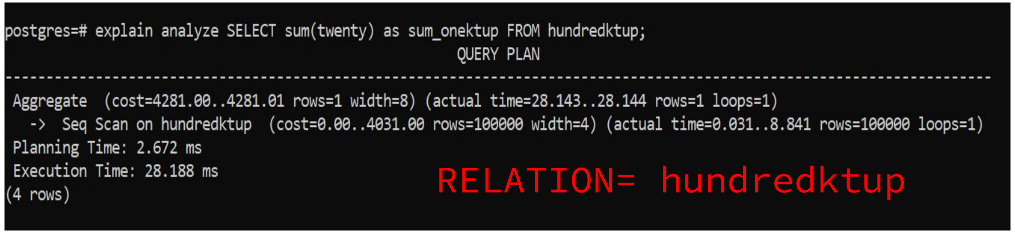
```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM tenktup1;
                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=429.00..429.01 rows=1 width=8) (actual time=2.781..2.782 rows=1 loops=1)
   ->  Seq Scan on tenktup1  (cost=0.00..404.00 rows=10000 width=4) (actual time=0.014..1.228 rows=10000 loops=1)
 Planning Time: 0.170 ms
 Execution Time: 2.855 ms
(4 rows)
```

RELATION= tenktup



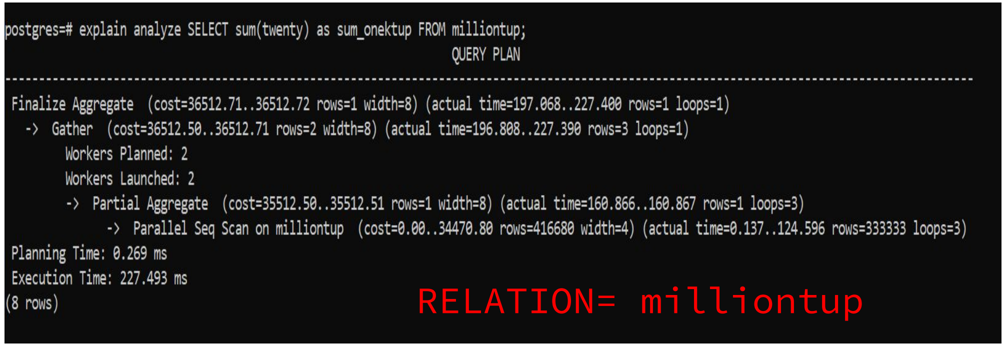
```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM fiftyktup;
                                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2141.00..2141.01 rows=1 width=8) (actual time=18.992..18.992 rows=1 loops=1)
   ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=4) (actual time=0.053..5.910 rows=50000 loops=1)
 Planning Time: 11.624 ms
 Execution Time: 19.037 ms
(4 rows)
```

RELATION= fiftyktup



```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM hundredktup;
                                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4281.00..4281.01 rows=1 width=8) (actual time=28.143..28.144 rows=1 loops=1)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=4) (actual time=0.031..8.841 rows=100000 loops=1)
 Planning Time: 2.672 ms
 Execution Time: 28.188 ms
(4 rows)
```

RELATION= hundredktup



```
postgres=# explain analyze SELECT sum(twenty) as sum_onektup FROM milliontup;
                                                   QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=36512.71..36512.72 rows=1 width=8) (actual time=197.068..227.400 rows=1 loops=1)
   ->  Gather  (cost=36512.50..36512.71 rows=2 width=8) (actual time=196.808..227.390 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=35512.50..35512.51 rows=1 width=8) (actual time=160.866..160.867 rows=1 loops=3)
               ->  Parallel Seq Scan on milliontup  (cost=0.00..34470.80 rows=416680 width=4) (actual time=0.137..124.596 rows=333333 loops=3)
 Planning Time: 0.269 ms
 Execution Time: 227.493 ms
(8 rows)
```

RELATION= milliontup

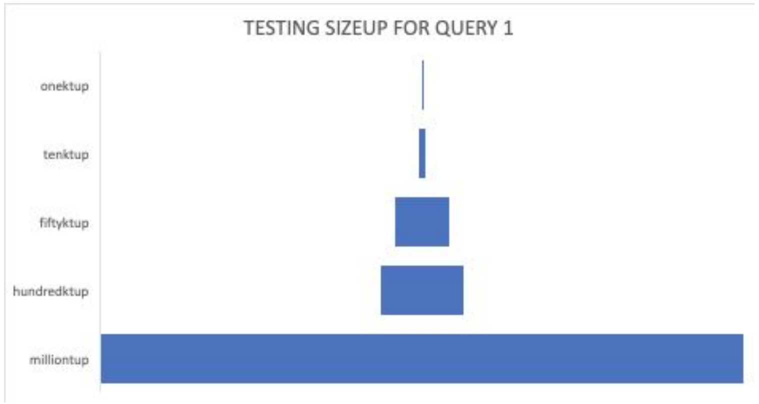
Experiment -4 Testing sizeup

QUERY 1 **SELECT SUM** (twenty) as sum_twenty

FROM onektup /tenktup / fiftyktup

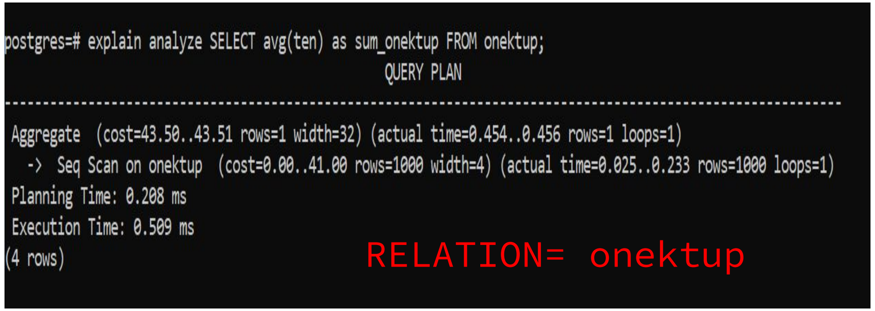
/hundredktup/ milliontup

Relation	onektup Execution Time (ms)	tenktup Execution Time (ms)	fiftyktup Execution Time (ms)	hundredktup Execution Time (ms)	milliontup Execution Time (ms)
1	0.715	2.855	19.037	28.188	227.493
2	0.383	2.991	19.558	34.718	216.907
3	0.377	3.355	14.877	33.271	222.260
4	0.381	2.809	16.104	30.225	218.913
5	0.360	2.557	14.137	27.130	216.133
Average	0.380	2.885	16.67	30.561	219.36



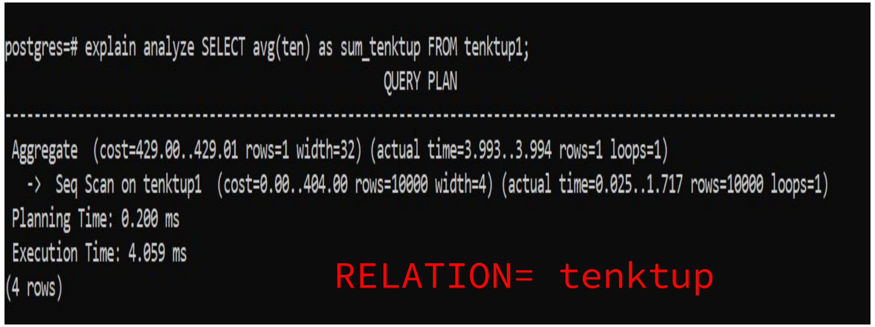
Experiment -4 Testing sizeup

QUERY 2 **SELECT AVG** (ten) as avg_ten
FROM onektup /tenktup / fiftyktup
/hundredktup/ milliontup



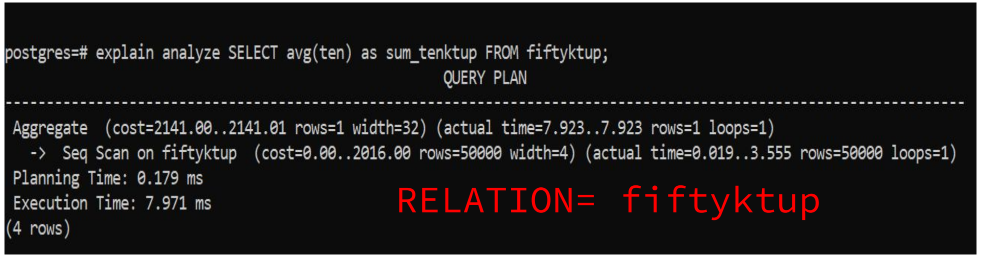
```
postgres=# explain analyze SELECT avg(ten) as sum_onektup FROM onektup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=43.50..43.51 rows=1 width=32) (actual time=0.454..0.456 rows=1 loops=1)
   ->  Seq Scan on onektup  (cost=0.00..41.00 rows=1000 width=4) (actual time=0.025..0.233 rows=1000 loops=1)
 Planning Time: 0.208 ms
 Execution Time: 0.509 ms
(4 rows)
```

RELATION= onektup



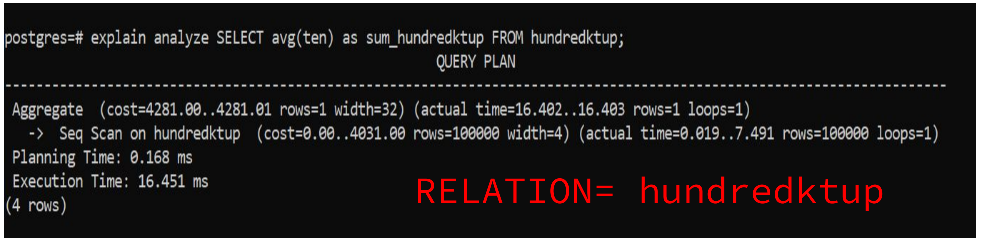
```
postgres=# explain analyze SELECT avg(ten) as sum_tenktup FROM tenktup1;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=429.00..429.01 rows=1 width=32) (actual time=3.993..3.994 rows=1 loops=1)
   ->  Seq Scan on tenktup1  (cost=0.00..404.00 rows=10000 width=4) (actual time=0.025..1.717 rows=10000 loops=1)
 Planning Time: 0.200 ms
 Execution Time: 4.059 ms
(4 rows)
```

RELATION= tenktup



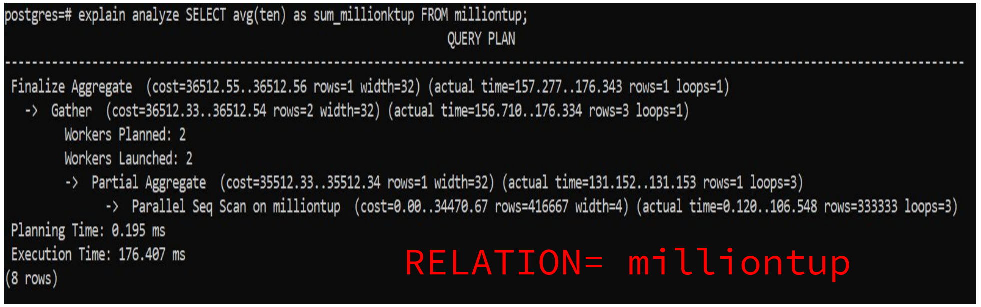
```
postgres=# explain analyze SELECT avg(ten) as sum_tenktup FROM fiftyktup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=2141.00..2141.01 rows=1 width=32) (actual time=7.923..7.923 rows=1 loops=1)
   ->  Seq Scan on fiftyktup  (cost=0.00..2016.00 rows=50000 width=4) (actual time=0.019..3.555 rows=50000 loops=1)
 Planning Time: 0.179 ms
 Execution Time: 7.971 ms
(4 rows)
```

RELATION= fiftyktup



```
postgres=# explain analyze SELECT avg(ten) as sum_hundredktup FROM hundredktup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4281.00..4281.01 rows=1 width=32) (actual time=16.402..16.403 rows=1 loops=1)
   ->  Seq Scan on hundredktup  (cost=0.00..4031.00 rows=100000 width=4) (actual time=0.019..7.491 rows=100000 loops=1)
 Planning Time: 0.168 ms
 Execution Time: 16.451 ms
(4 rows)
```

RELATION= hundredktup



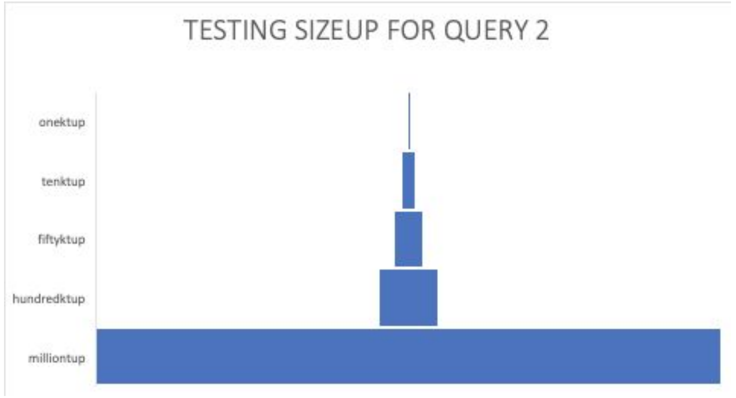
```
postgres=# explain analyze SELECT avg(ten) as sum_milliontup FROM milliontup;
                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=36512.55..36512.56 rows=1 width=32) (actual time=157.277..176.343 rows=1 loops=1)
   ->  Gather  (cost=36512.33..36512.54 rows=2 width=32) (actual time=156.710..176.334 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=35512.33..35512.34 rows=1 width=32) (actual time=131.152..131.153 rows=1 loops=3)
               ->  Parallel Seq Scan on milliontup  (cost=0.00..34470.67 rows=416667 width=4) (actual time=0.120..106.548 rows=333333 loops=3)
 Planning Time: 0.195 ms
 Execution Time: 176.407 ms
(8 rows)
```

RELATION= milliontup

Experiment -4 Testing sizeup

QUERY 2 **SELECT AVG** (ten) as avg_ten
FROM onektup /tenktup / fiftyktup
/hundredktup/ milliontup

Relation	onektup Execution Time (ms)	tenktup Execution Time (ms)	fiftyktup Execution Time (ms)	hundredktup Execution Time (ms)	milliontup Execution Time (ms)
1	0.509	4.059	7.971	16.451	176.407
2	0.693	4.048	8.828	13.965	185.505
3	0.460	4.051	9.313	13.224	175.801
4	0.578	4.546	8.745	16.442	178.247
5	0.548	3.468	7.862	14.373	179.302
Average	0.55	4.052	8.52	14.926	177.78



Thank You ！