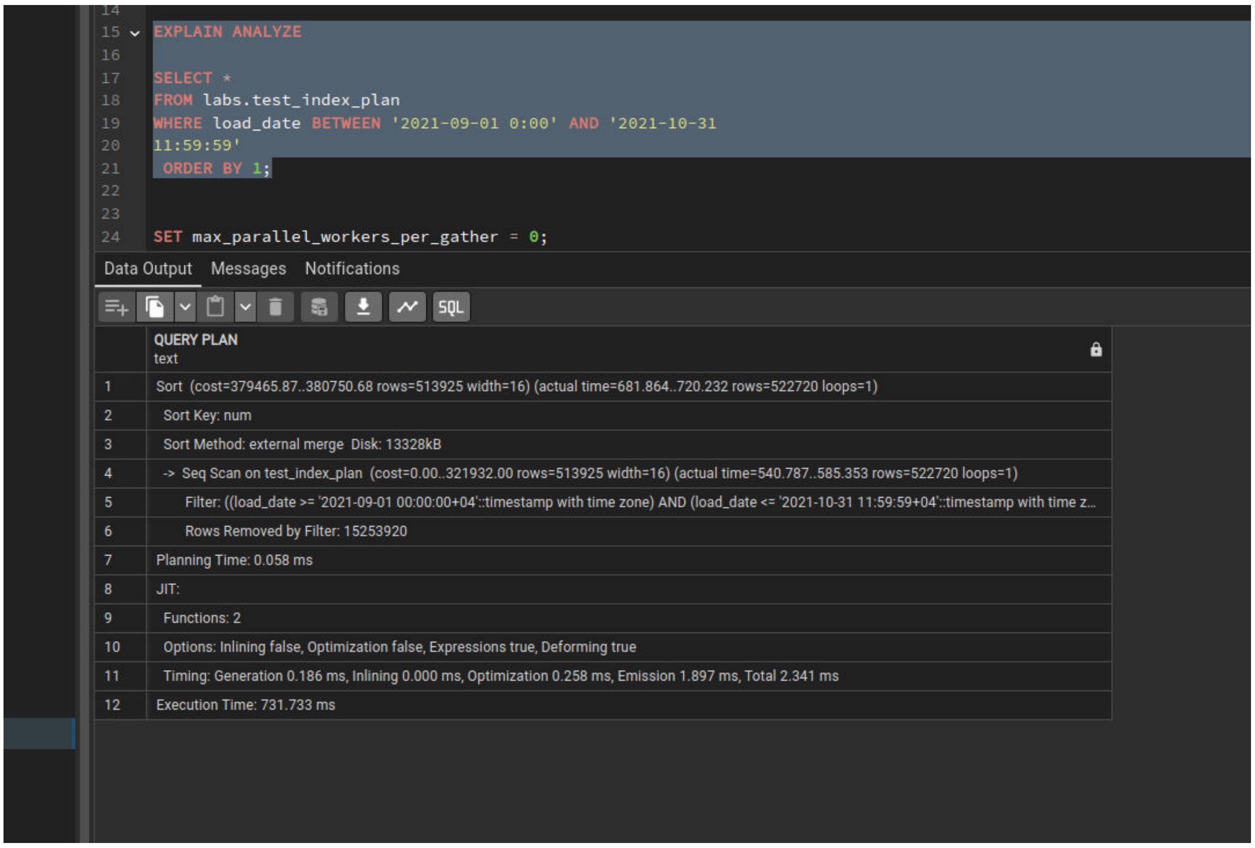
PG_DWH TASK 4

1.

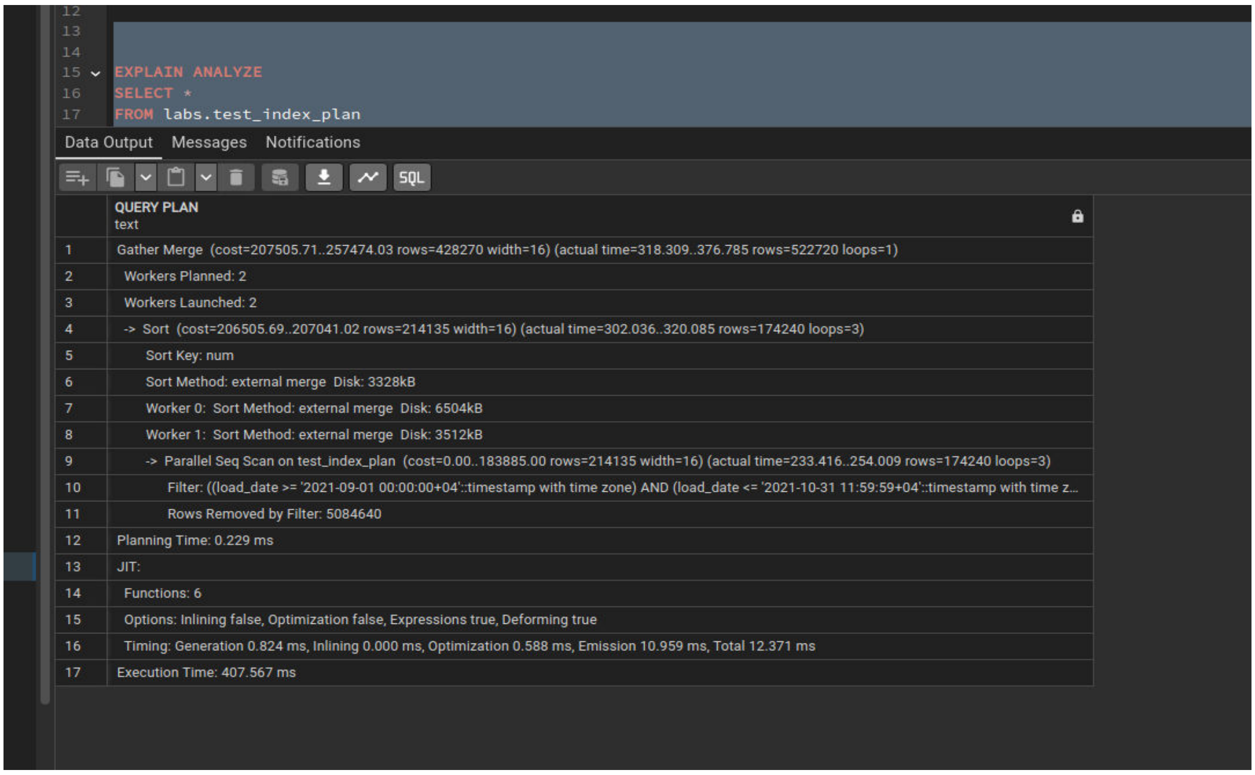


```sql
EXPLAIN ANALYZE

SELECT *
FROM labs.test_index_plan
WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31
11:59:59'
 ORDER BY 1;


SET max_parallel_workers_per_gather = 0;
```

Data Output | Messages | Notifications

```
QUERY PLAN
text
1   Sort  (cost=379465.87..380750.68 rows=513925 width=16) (actual time=681.864..720.232 rows=522720 loops=1)
2     Sort Key: num
3     Sort Method: external merge  Disk: 13328kB
4     ->  Seq Scan on test_index_plan  (cost=0.00..321932.00 rows=513925 width=16) (actual time=540.787..585.353 rows=522720 loops=1)
5           Filter: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time z...
6           Rows Removed by Filter: 15253920
7   Planning Time: 0.058 ms
8   JIT:
9     Functions: 2
10    Options: Inlining false, Optimization false, Expressions true, Deforming true
11    Timing: Generation 0.186 ms, Inlining 0.000 ms, Optimization 0.258 ms, Emission 1.897 ms, Total 2.341 ms
12  Execution Time: 731.733 ms
```

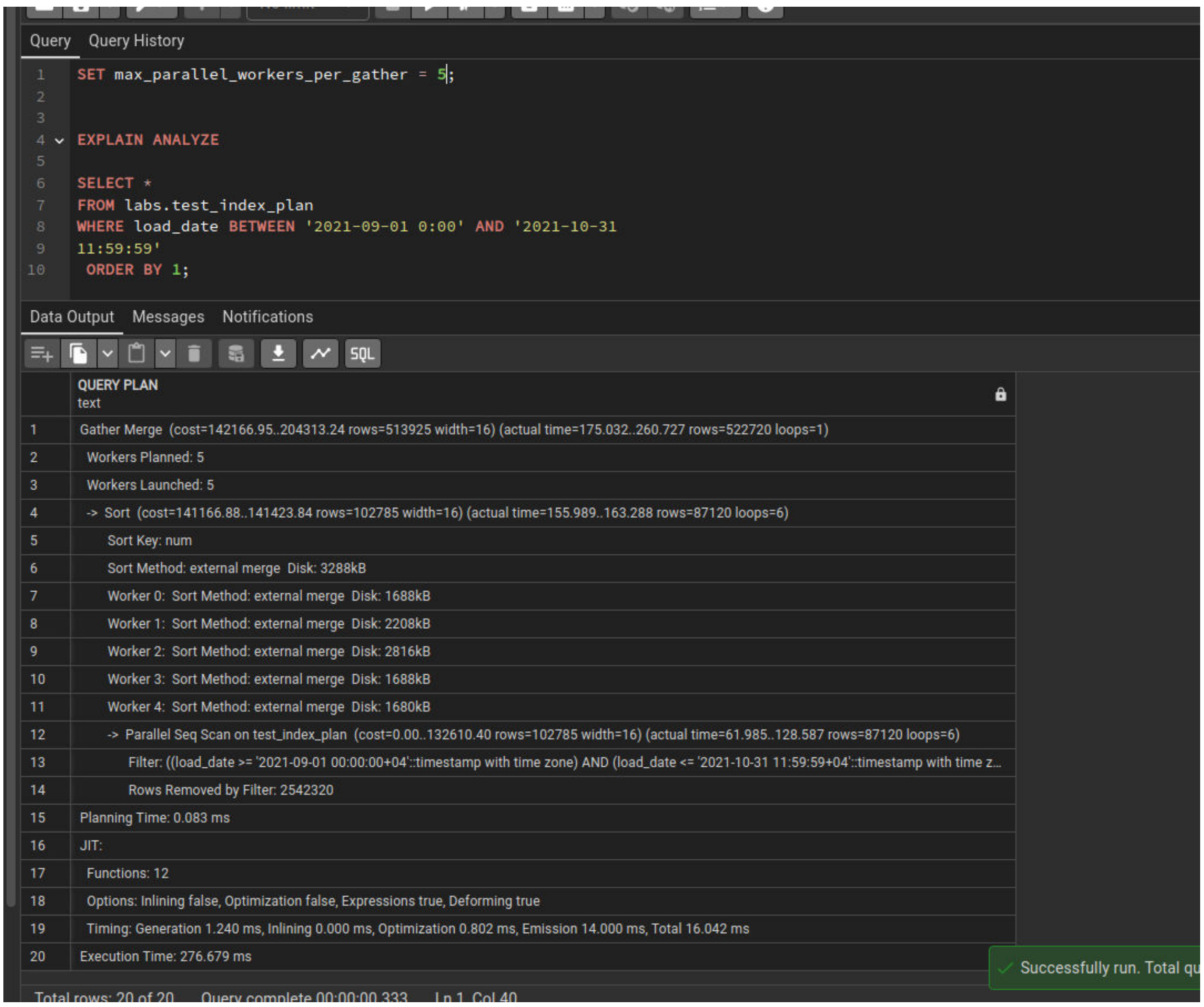
Here is the common query plan for our select statement to be executed , as we see the here is no information about workers but when we see more detailed query plan ,it shows that by default postgres decided to use 2 workers for this query to execute .We can set worker count up to 5.



```sql
EXPLAIN ANALYZE
SELECT *
FROM labs.test_index_plan
```

Data Output | Messages | Notifications

```
QUERY PLAN
text
1   Gather Merge  (cost=207505.71..257474.03 rows=428270 width=16) (actual time=318.309..376.785 rows=522720 loops=1)
2     Workers Planned: 2
3     Workers Launched: 2
4     ->  Sort  (cost=206505.69..207041.02 rows=214135 width=16) (actual time=302.036..320.085 rows=174240 loops=3)
5           Sort Key: num
6           Sort Method: external merge  Disk: 3328kB
7           Worker 0:  Sort Method: external merge  Disk: 6504kB
8           Worker 1:  Sort Method: external merge  Disk: 3512kB
9           ->  Parallel Seq Scan on test_index_plan  (cost=0.00..183885.00 rows=214135 width=16) (actual time=233.416..254.009 rows=174240 loops=3)
10                Filter: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time z...
11                Rows Removed by Filter: 5084640
12  Planning Time: 0.229 ms
13  JIT:
14    Functions: 6
15    Options: Inlining false, Optimization false, Expressions true, Deforming true
16    Timing: Generation 0.824 ms, Inlining 0.000 ms, Optimization 0.588 ms, Emission 10.959 ms, Total 12.371 ms
17  Execution Time: 407.567 ms
```


Query Query History

```sql
SET max_parallel_workers_per_gather = 5;


EXPLAIN ANALYZE

SELECT *
FROM labs.test_index_plan
WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31
11:59:59'
 ORDER BY 1;
```

Data Output Messages Notifications

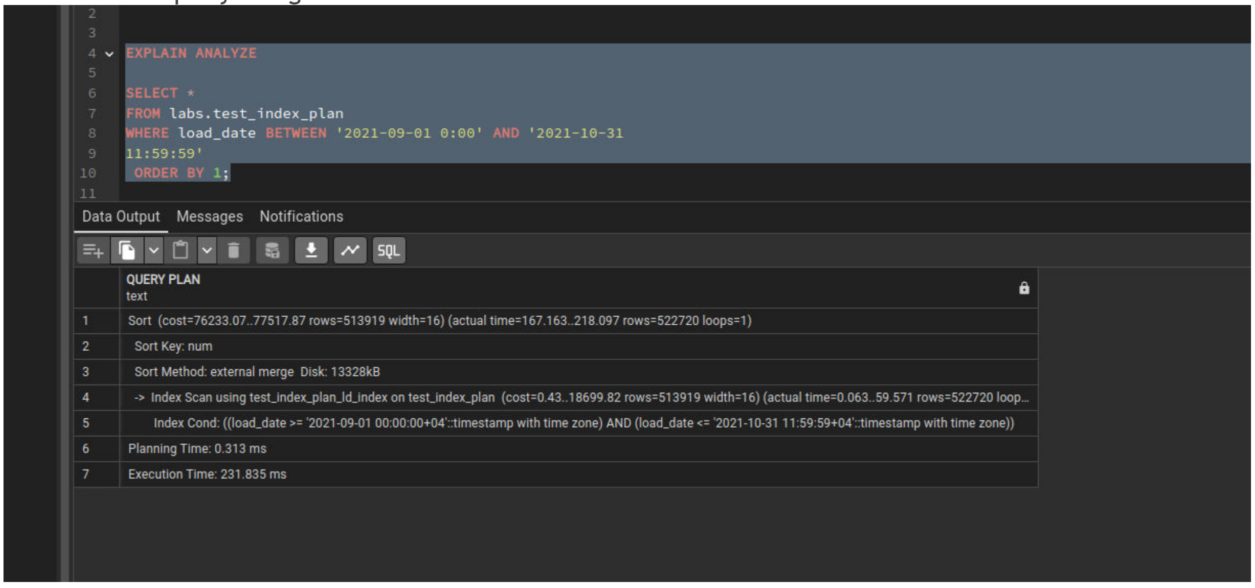
	QUERY PLAN text
1	Gather Merge (cost=142166.95..204313.24 rows=513925 width=16) (actual time=175.032..260.727 rows=522720 loops=1)
2	Workers Planned: 5
3	Workers Launched: 5
4	-> Sort (cost=141166.88..141423.84 rows=102785 width=16) (actual time=155.989..163.288 rows=87120 loops=6)
5	Sort Key: num
6	Sort Method: external merge Disk: 3288kB
7	Worker 0: Sort Method: external merge Disk: 1688kB
8	Worker 1: Sort Method: external merge Disk: 2208kB
9	Worker 2: Sort Method: external merge Disk: 2816kB
10	Worker 3: Sort Method: external merge Disk: 1688kB
11	Worker 4: Sort Method: external merge Disk: 1680kB
12	-> Parallel Seq Scan on test_index_plan (cost=0.00..132610.40 rows=102785 width=16) (actual time=61.985..128.587 rows=87120 loops=6)
13	Filter: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time z...
14	Rows Removed by Filter: 2542320
15	Planning Time: 0.083 ms
16	JIT:
17	Functions: 12
18	Options: Inlining false, Optimization false, Expressions true, Deforming true
19	Timing: Generation 1.240 ms, Inlining 0.000 ms, Optimization 0.802 ms, Emission 14.000 ms, Total 16.042 ms
20	Execution Time: 276.679 ms

Successfully run. Total qu

Total rows: 20 of 20 Query complete 00:00:00.333 Ln 1, Col 40

Here you can see that this time it executed with 5 workers.

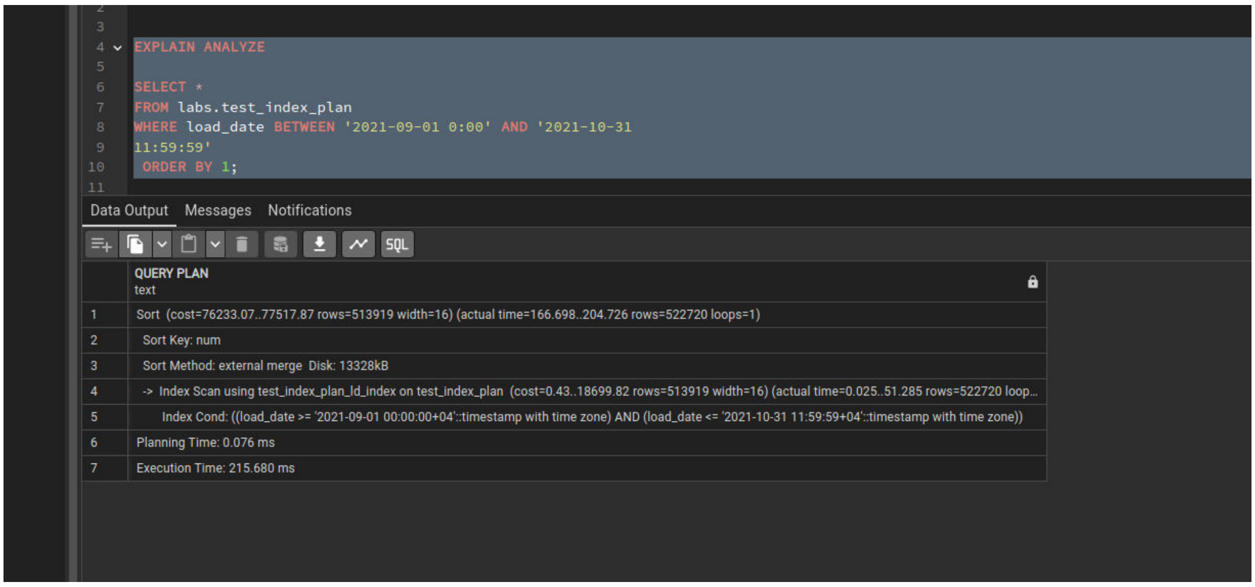
2. After created INDEX on the table test_index_plan we see that query plan has been changed and now it's executes query using index scan.


```sql
EXPLAIN ANALYZE

SELECT *
FROM labs.test_index_plan
WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31
11:59:59'
 ORDER BY 1;
```

Data Output Messages Notifications

	QUERY PLAN text
1	Sort (cost=76233.07..77517.87 rows=513919 width=16) (actual time=167.163..218.097 rows=522720 loops=1)
2	Sort Key: num
3	Sort Method: external merge Disk: 13328kB
4	-> Index Scan using test_index_plan_ld_index on test_index_plan (cost=0.43..18699.82 rows=513919 width=16) (actual time=0.063..59.571 rows=522720 loop...
5	Index Cond: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time zone))
6	Planning Time: 0.313 ms
7	Execution Time: 231.835 ms

1

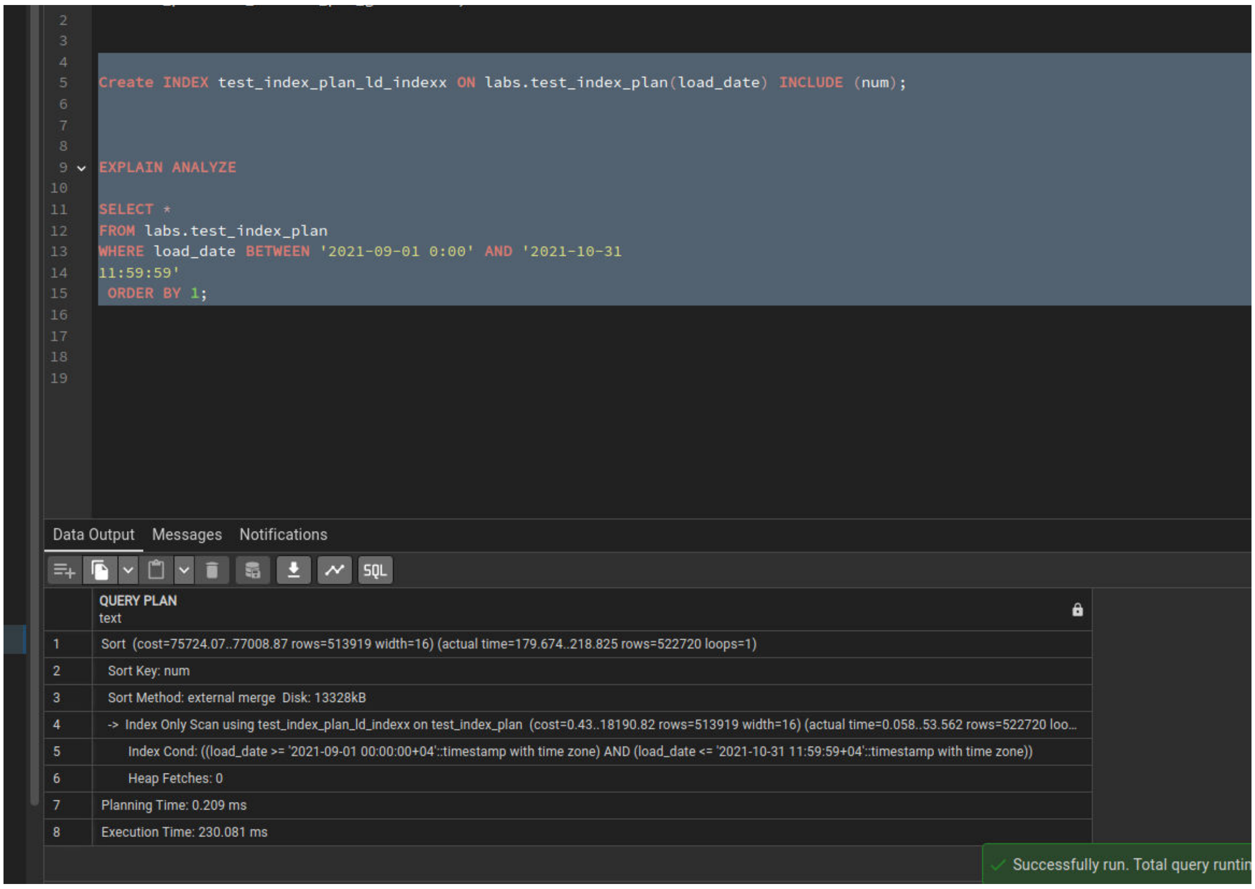


```sql
EXPLAIN ANALYZE

SELECT *
FROM labs.test_index_plan
WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31
11:59:59'
 ORDER BY 1;
```

	QUERY PLAN text
1	Sort (cost=76233.07..77517.87 rows=513919 width=16) (actual time=166.698..204.726 rows=522720 loops=1)
2	Sort Key: num
3	Sort Method: external merge Disk: 13328kB
4	-> Index Scan using test_index_plan_ld_index on test_index_plan (cost=0.43..18699.82 rows=513919 width=16) (actual time=0.025..51.285 rows=522720 loop...
5	Index Cond: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time zone))
6	Planning Time: 0.076 ms
7	Execution Time: 215.680 ms

After disabled parallel workers it’s still uses created index to execute the query.

To use INDEX ONLY SCAN method we can created INDEX and include olso other columns .

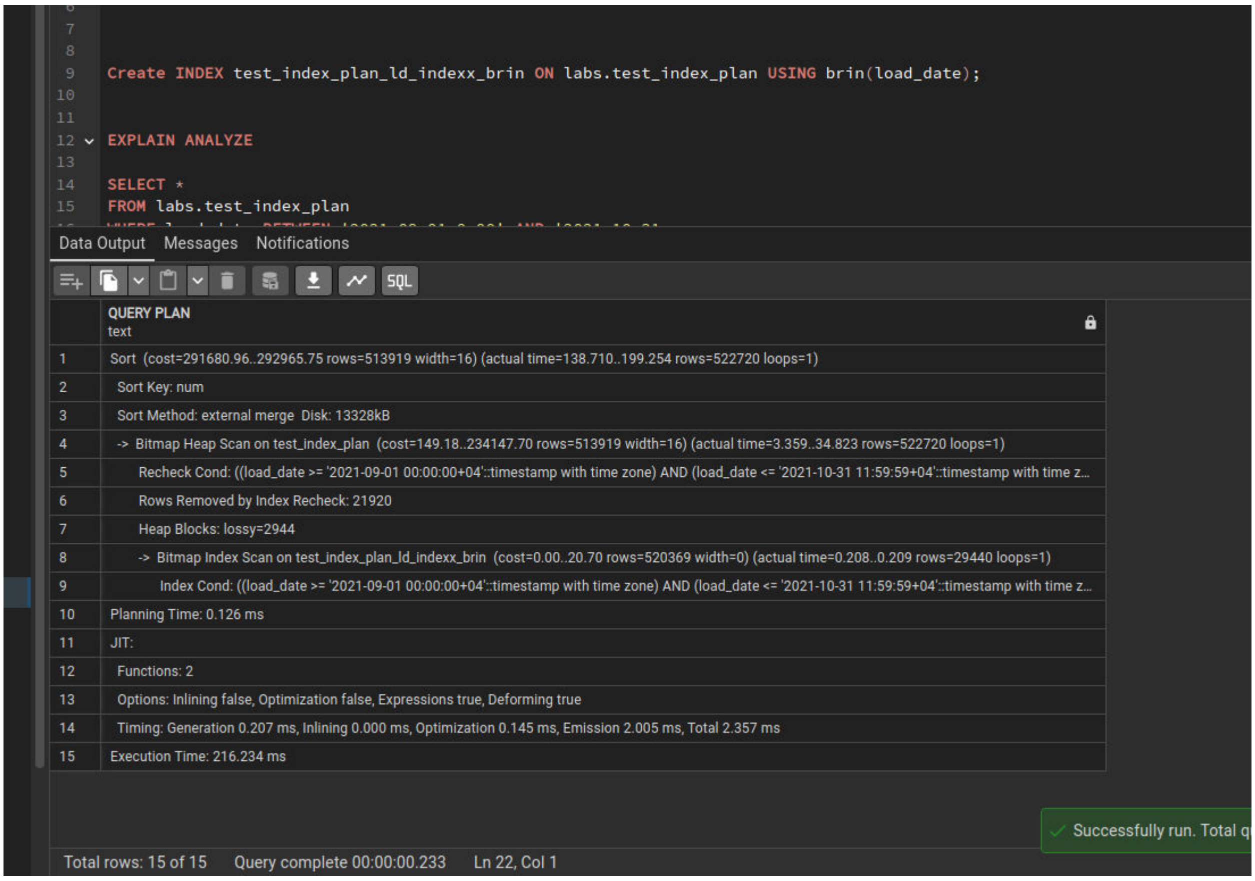


```sql
Create INDEX test_index_plan_ld_indexx ON labs.test_index_plan(load_date) INCLUDE (num);



EXPLAIN ANALYZE

SELECT *
FROM labs.test_index_plan
WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31
11:59:59'
 ORDER BY 1;
```

	QUERY PLAN text
1	Sort (cost=75724.07..77008.87 rows=513919 width=16) (actual time=179.674..218.825 rows=522720 loops=1)
2	Sort Key: num
3	Sort Method: external merge Disk: 13328kB
4	-> Index Only Scan using test_index_plan_ld_indexx on test_index_plan (cost=0.43..18190.82 rows=513919 width=16) (actual time=0.058..53.562 rows=522720 loo...
5	Index Cond: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time zone))
6	Heap Fetches: 0
7	Planning Time: 0.209 ms
8	Execution Time: 230.081 ms

Successfully run. Total query runtin

By creating BRIN INDEX for our table and inspecting the query plan we notice that Postgres will use Bitmap Heap Scan to execute the query.

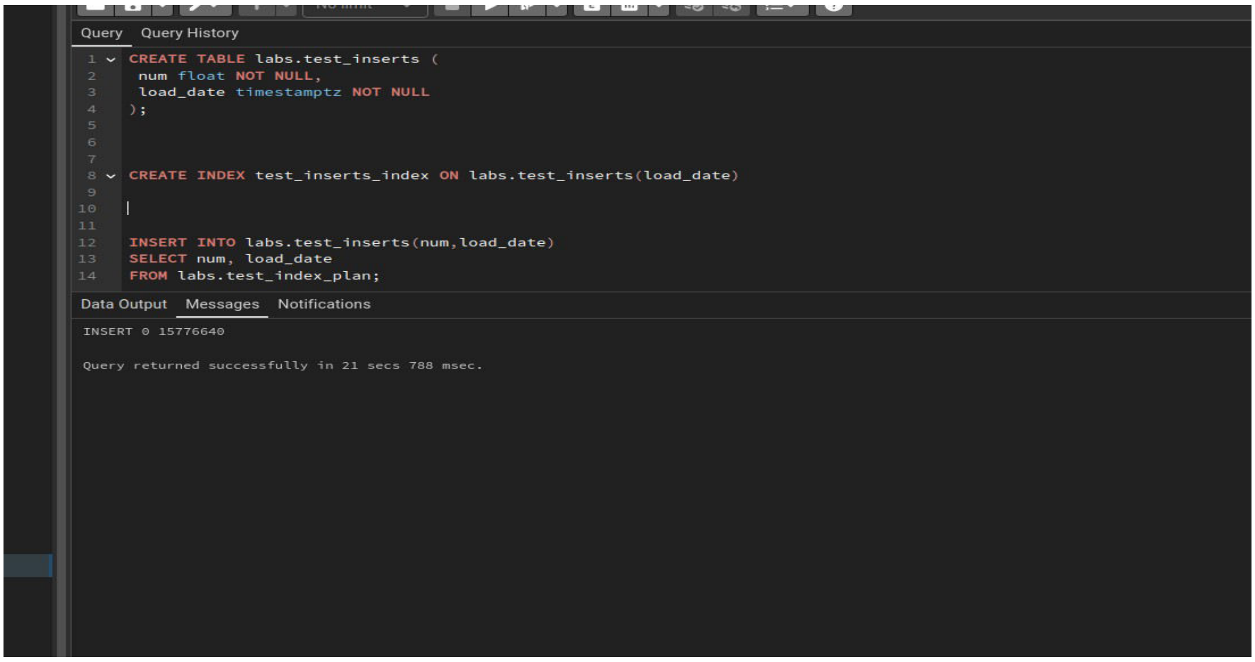


```
Create INDEX test_index_plan_ld_indexx_brin ON labs.test_index_plan USING brin(load_date);


EXPLAIN ANALYZE

SELECT *
FROM labs.test_index_plan
```

	QUERY PLAN text
1	Sort (cost=291680.96..292965.75 rows=513919 width=16) (actual time=138.710..199.254 rows=522720 loops=1)
2	Sort Key: num
3	Sort Method: external merge Disk: 13328kB
4	-> Bitmap Heap Scan on test_index_plan (cost=149.18..234147.70 rows=513919 width=16) (actual time=3.359..34.823 rows=522720 loops=1)
5	Recheck Cond: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time z...
6	Rows Removed by Index Recheck: 21920
7	Heap Blocks: lossy=2944
8	-> Bitmap Index Scan on test_index_plan_ld_indexx_brin (cost=0.00..20.70 rows=520369 width=0) (actual time=0.208..0.209 rows=29440 loops=1)
9	Index Cond: ((load_date >= '2021-09-01 00:00:00+04'::timestamp with time zone) AND (load_date <= '2021-10-31 11:59:59+04'::timestamp with time z...
10	Planning Time: 0.126 ms
11	JIT:
12	Functions: 2
13	Options: Inlining false, Optimization false, Expressions true, Deforming true
14	Timing: Generation 0.207 ms, Inlining 0.000 ms, Optimization 0.145 ms, Emission 2.005 ms, Total 2.357 ms
15	Execution Time: 216.234 ms

```
Total rows: 15 of 15    Query complete 00:00:00.233    Ln 22, Col 1
```

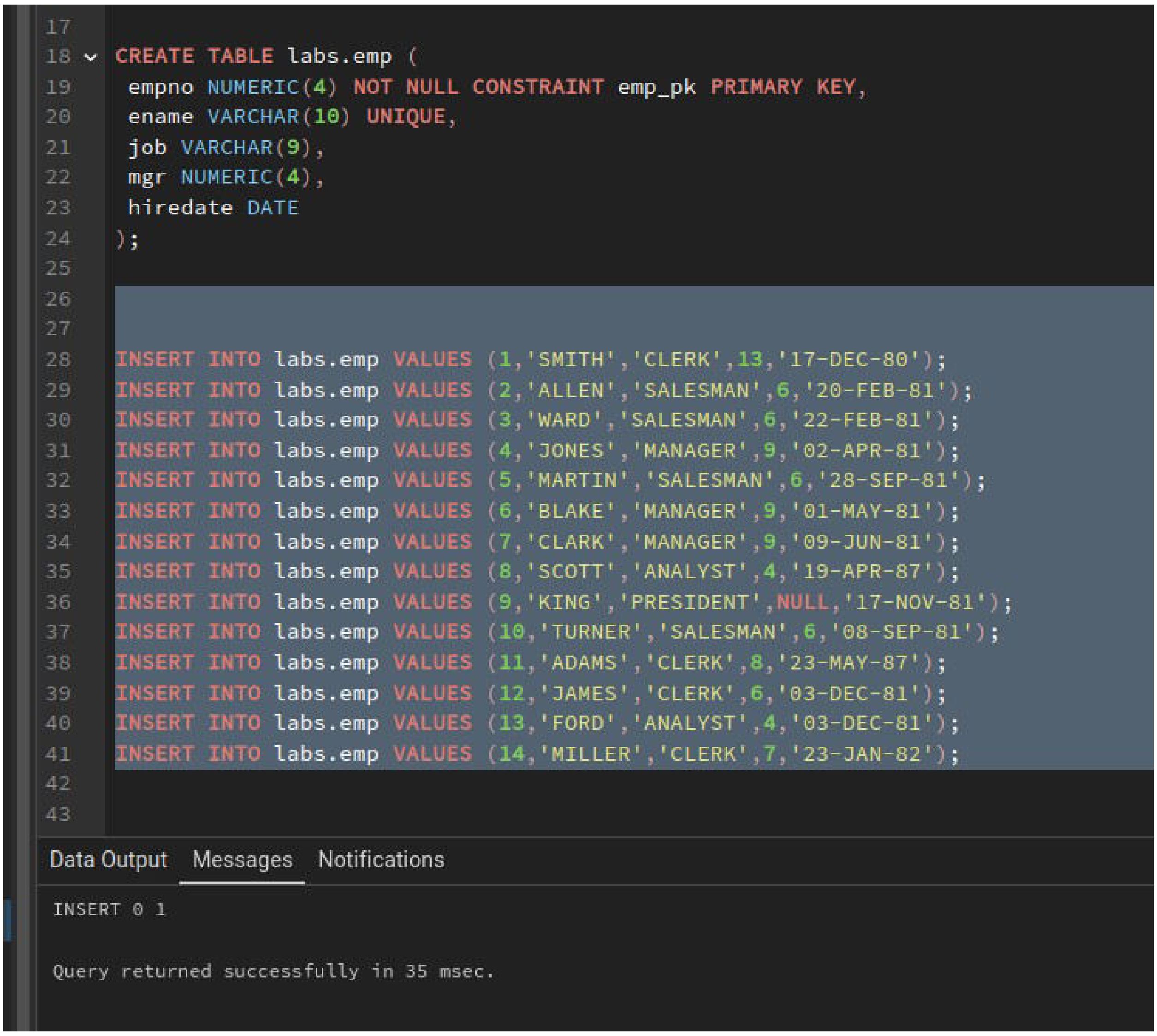
2. Table test_inserts created , index created and all data from test_index_plan table inserted to this one.



```
CREATE TABLE labs.test_inserts (
 num float NOT NULL,
 load_date timestamptz NOT NULL
);



CREATE INDEX test_inserts_index ON labs.test_inserts(load_date)



INSERT INTO labs.test_inserts(num,load_date)
SELECT num, load_date
FROM labs.test_index_plan;
```

```
INSERT 0 15776640

Query returned successfully in 21 secs 788 msec.
```

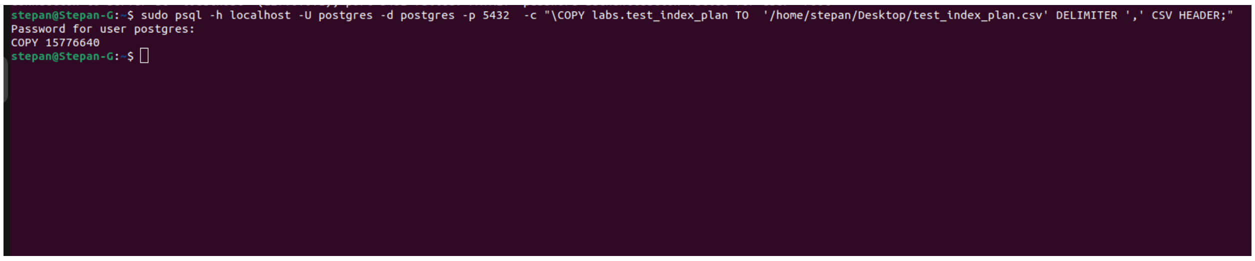
Values inserted into table.



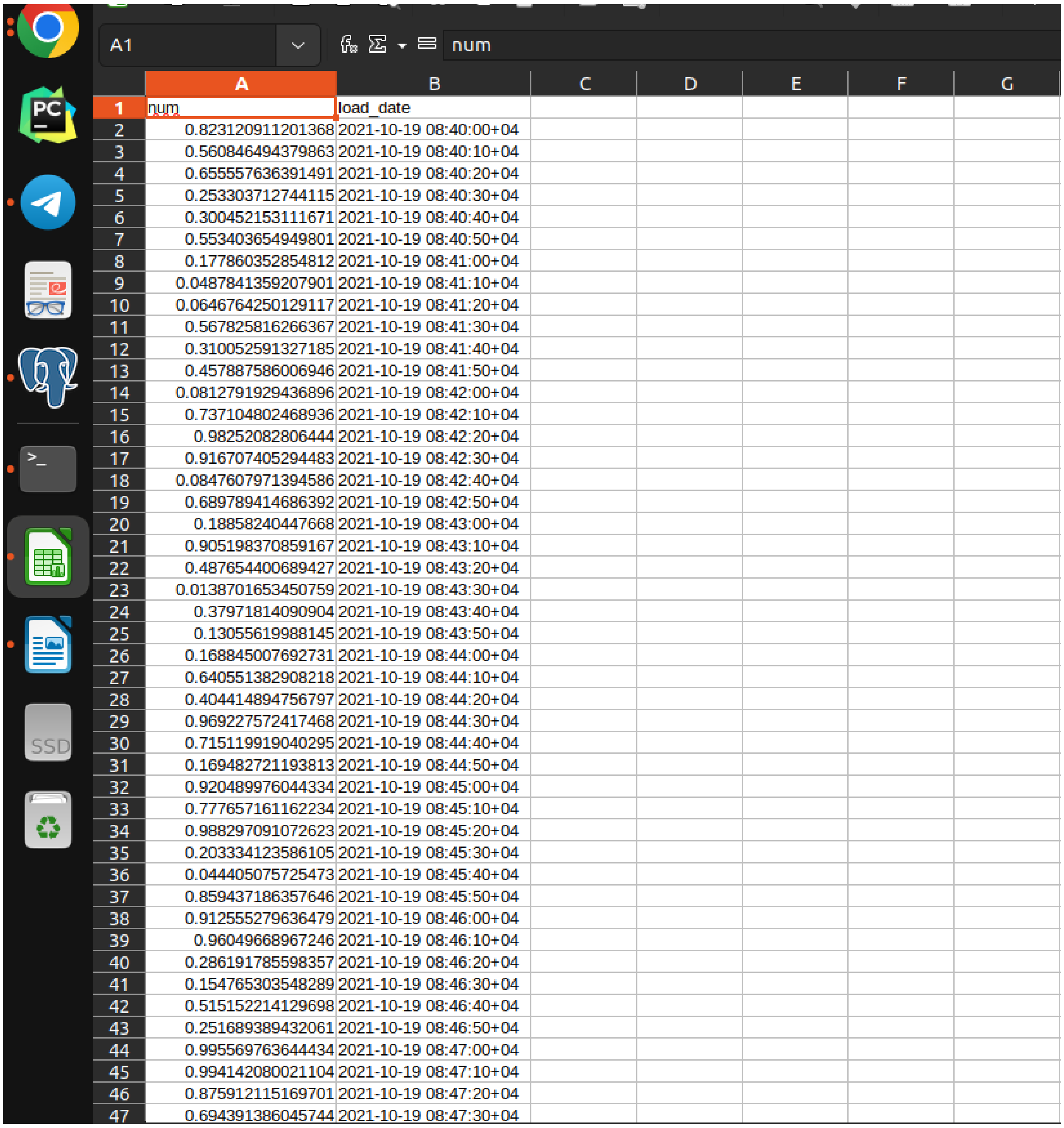
```
CREATE TABLE labs.emp (
 empno NUMERIC(4) NOT NULL CONSTRAINT emp_pk PRIMARY KEY,
 ename VARCHAR(10) UNIQUE,
 job VARCHAR(9),
 mgr NUMERIC(4),
 hiredate DATE
);



INSERT INTO labs.emp VALUES (1,'SMITH','CLERK',13,'17-DEC-80');
INSERT INTO labs.emp VALUES (2,'ALLEN','SALESMAN',6,'20-FEB-81');
INSERT INTO labs.emp VALUES (3,'WARD','SALESMAN',6,'22-FEB-81');
INSERT INTO labs.emp VALUES (4,'JONES','MANAGER',9,'02-APR-81');
INSERT INTO labs.emp VALUES (5,'MARTIN','SALESMAN',6,'28-SEP-81');
INSERT INTO labs.emp VALUES (6,'BLAKE','MANAGER',9,'01-MAY-81');
INSERT INTO labs.emp VALUES (7,'CLARK','MANAGER',9,'09-JUN-81');
INSERT INTO labs.emp VALUES (8,'SCOTT','ANALYST',4,'19-APR-87');
INSERT INTO labs.emp VALUES (9,'KING','PRESIDENT',NULL,'17-NOV-81');
INSERT INTO labs.emp VALUES (10,'TURNER','SALESMAN',6,'08-SEP-81');
INSERT INTO labs.emp VALUES (11,'ADAMS','CLERK',8,'23-MAY-87');
INSERT INTO labs.emp VALUES (12,'JAMES','CLERK',6,'03-DEC-81');
INSERT INTO labs.emp VALUES (13,'FORD','ANALYST',4,'03-DEC-81');
INSERT INTO labs.emp VALUES (14,'MILLER','CLERK',7,'23-JAN-82');
```

```
INSERT 0 1

Query returned successfully in 35 msec.
```

When I tried to COPY table into CSV from pgadmin it show’s permission denied error , I was able to execute this query in psql in terminal.



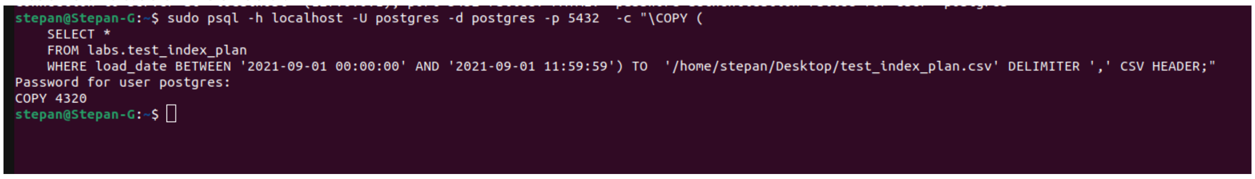
```
stepan@Stepan-G:~$ sudo psql -h localhost -U postgres -d postgres -p 5432  -c "\COPY labs.test_index_plan TO  '/home/stepan/Desktop/test_index_plan.csv' DELIMITER ',' CSV HEADER;"
Password for user postgres:
COPY 15776640
stepan@Stepan-G:~$ 
```



A1 num

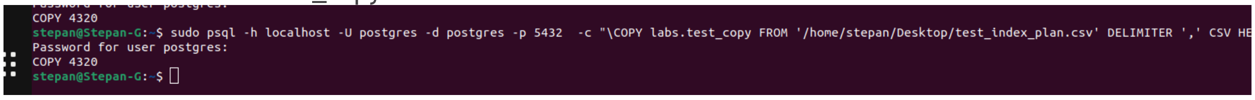
	A	B	C	D	E	F	G
1	num	load_date					
2	0.823120911201368	2021-10-19 08:40:00+04					
3	0.560846494379863	2021-10-19 08:40:10+04					
4	0.655557636391491	2021-10-19 08:40:20+04					
5	0.253303712744115	2021-10-19 08:40:30+04					
6	0.300452153111671	2021-10-19 08:40:40+04					
7	0.553403654949801	2021-10-19 08:40:50+04					
8	0.177860352854812	2021-10-19 08:41:00+04					
9	0.0487841359207901	2021-10-19 08:41:10+04					
10	0.0646764250129117	2021-10-19 08:41:20+04					
11	0.567825816266367	2021-10-19 08:41:30+04					
12	0.310052591327185	2021-10-19 08:41:40+04					
13	0.457887586006946	2021-10-19 08:41:50+04					
14	0.0812791929436896	2021-10-19 08:42:00+04					
15	0.737104802468936	2021-10-19 08:42:10+04					
16	0.98252082806444	2021-10-19 08:42:20+04					
17	0.916707405294483	2021-10-19 08:42:30+04					
18	0.0847607971394586	2021-10-19 08:42:40+04					
19	0.689789414686392	2021-10-19 08:42:50+04					
20	0.18858240447668	2021-10-19 08:43:00+04					
21	0.905198370859167	2021-10-19 08:43:10+04					
22	0.487654400689427	2021-10-19 08:43:20+04					
23	0.0138701653450759	2021-10-19 08:43:30+04					
24	0.37971814090904	2021-10-19 08:43:40+04					
25	0.13055619988145	2021-10-19 08:43:50+04					
26	0.168845007692731	2021-10-19 08:44:00+04					
27	0.640551382908218	2021-10-19 08:44:10+04					
28	0.404414894756797	2021-10-19 08:44:20+04					
29	0.969227572417468	2021-10-19 08:44:30+04					
30	0.715119919040295	2021-10-19 08:44:40+04					
31	0.169482721193813	2021-10-19 08:44:50+04					
32	0.920489976044334	2021-10-19 08:45:00+04					
33	0.777657161162234	2021-10-19 08:45:10+04					
34	0.988297091072623	2021-10-19 08:45:20+04					
35	0.203334123586105	2021-10-19 08:45:30+04					
36	0.044405075725473	2021-10-19 08:45:40+04					
37	0.859437186357646	2021-10-19 08:45:50+04					
38	0.912555279636479	2021-10-19 08:46:00+04					
39	0.96049668967246	2021-10-19 08:46:10+04					
40	0.286191785598357	2021-10-19 08:46:20+04					
41	0.154765303548289	2021-10-19 08:46:30+04					
42	0.515152214129698	2021-10-19 08:46:40+04					
43	0.251689389432061	2021-10-19 08:46:50+04					
44	0.995569763644434	2021-10-19 08:47:00+04					
45	0.994142080021104	2021-10-19 08:47:10+04					
46	0.875912115169701	2021-10-19 08:47:20+04					
47	0.694391386045744	2021-10-19 08:47:30+04					

Filtered dated inserted.



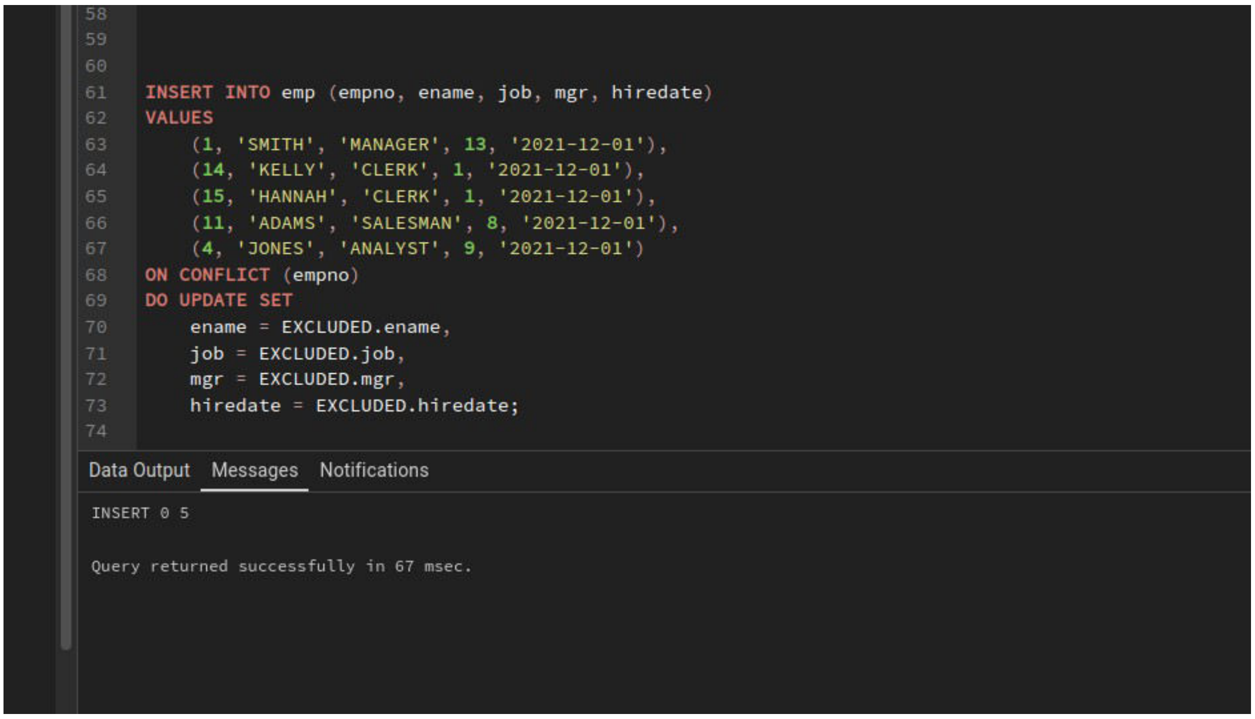
```
stepan@Stepan-G:~$ sudo psql -h localhost -U postgres -d postgres -p 5432  -c "\COPY (
    SELECT *
    FROM labs.test_index_plan
    WHERE load_date BETWEEN '2021-09-01 00:00:00' AND '2021-09-01 11:59:59') TO  '/home/stepan/Desktop/test_index_plan.csv' DELIMITER ',' CSV HEADER;"
Password for user postgres:
COPY 4320
stepan@Stepan-G:~$ 
```

All data inserted into test_copy table.



```
COPY 4320
stepan@Stepan-G:~$ sudo psql -h localhost -U postgres -d postgres -p 5432  -c "\COPY labs.test_copy FROM '/home/stepan/Desktop/test_index_plan.csv' DELIMITER ',' CSV HE
Password for user postgres:
COPY 4320
stepan@Stepan-G:~$ 
```

UPSERT statement for emp table , when it’s get uniqueness error on column empno then it’s going to update that row with provided values , id remians same.



```
INSERT INTO emp (empno, ename, job, mgr, hiredate)
VALUES
    (1, 'SMITH', 'MANAGER', 13, '2021-12-01'),
    (14, 'KELLY', 'CLERK', 1, '2021-12-01'),
    (15, 'HANNAH', 'CLERK', 1, '2021-12-01'),
    (11, 'ADAMS', 'SALESMAN', 8, '2021-12-01'),
    (4, 'JONES', 'ANALYST', 9, '2021-12-01')
ON CONFLICT (empno)
DO UPDATE SET
    ename = EXCLUDED.ename,
    job = EXCLUDED.job,
    mgr = EXCLUDED.mgr,
    hiredate = EXCLUDED.hiredate;
```

```
INSERT 0 5

Query returned successfully in 67 msec.
```