DATABASE MANAGEMENT SYSTEM

AIRLINES RESERVATION SYSTEM

TEAM ID: 05

RITHIKA PAI
PES1UG19CS388

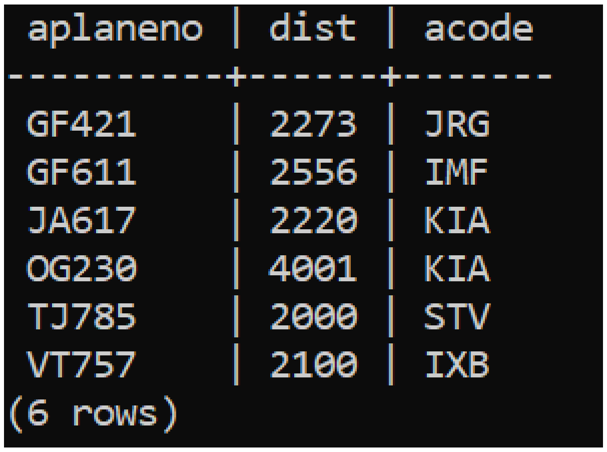
RITHIKA SHANKAR
PES1UG19CS387

RAMYA N PRABHU
PES1UG19CS380

Simple Queries

- **All the flights that fly more than 1500 kms**

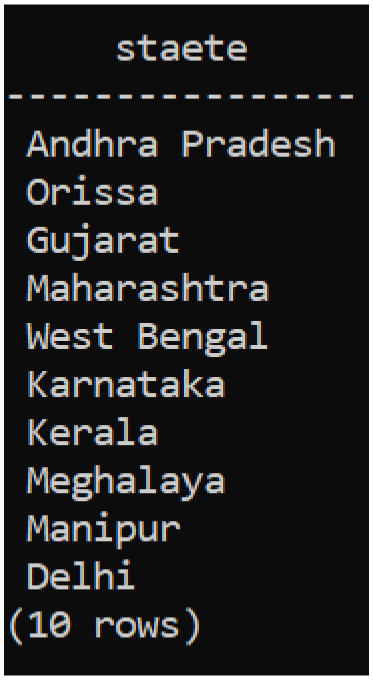
```
select distinct on (aplaneno) aplaneno, dist, acode from stops_at where dist>1500;
```



```
 aplaneno | dist | acode
----------+------+-------
 GF421    | 2273 | JRG
 GF611    | 2556 | IMF
 JA617    | 2220 | KIA
 OG230    | 4001 | KIA
 TJ785    | 2000 | STV
 VT757    | 2100 | IXB
(6 rows)
```

- **All the states whose airports are in the system**

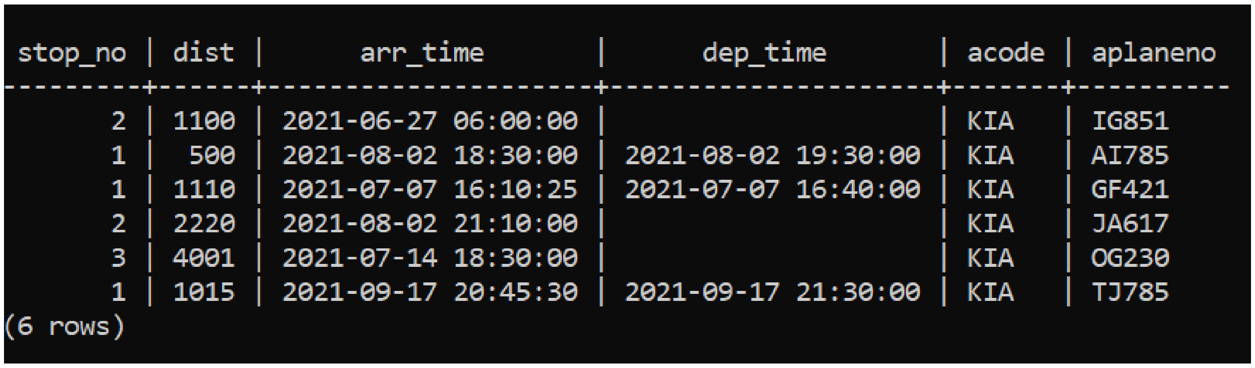
```
select staete from airport group by staete;
```



```
     staete
-----------------
 Andhra Pradesh
 Orissa
 Gujarat
 Maharashtra
 West Bengal
 Karnataka
 Kerala
 Meghalaya
 Manipur
 Delhi
(10 rows)
```

- **Flight details for all flights flying to bangalore**

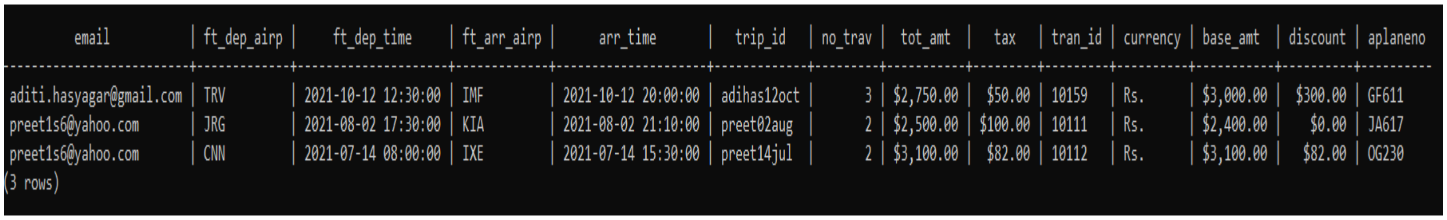
```
select * from stops_at where acode='KIA';
```



```
 stop_no | dist |      arr_time       |      dep_time       | acode | aplaneno
---------+------+---------------------+---------------------+-------+----------
       2 | 1100 | 2021-06-27 06:00:00 |                     | KIA   | IG851
       1 |  500 | 2021-08-02 18:30:00 | 2021-08-02 19:30:00 | KIA   | AI785
       1 | 1110 | 2021-07-07 16:10:25 | 2021-07-07 16:40:00 | KIA   | GF421
       2 | 2220 | 2021-08-02 21:10:00 |                     | KIA   | JA617
       3 | 4001 | 2021-07-14 18:30:00 |                     | KIA   | OG230
       1 | 1015 | 2021-09-17 20:45:30 | 2021-09-17 21:30:00 | KIA   | TJ785
(6 rows)
```

- **Trips where there are more than 1 travellers**

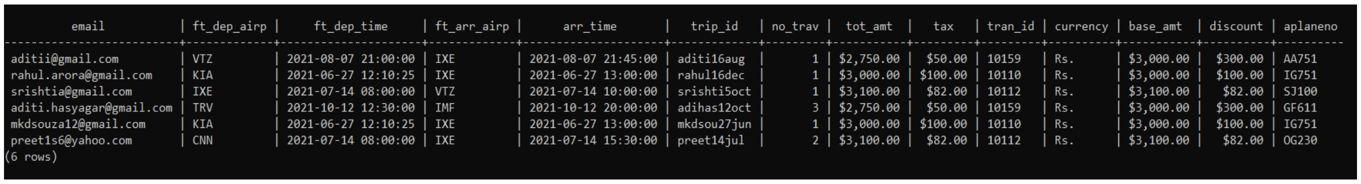
```
select * from flight_trip where no_trav>1;
```



```
         email          | ft_dep_airp |     ft_dep_time     | ft_arr_airp |      arr_time       |   trip_id   | no_trav |  tot_amt  |   tax   | tran_id | currency | base_amt  | discount | aplaneno
------------------------+-------------+---------------------+-------------+---------------------+-------------+---------+-----------+---------+---------+----------+-----------+----------+---------
aditi.hasyagar@gmail.com | TRV        | 2021-10-12 12:30:00 | IMF         | 2021-10-12 20:00:00 | adihas12oct |       3 | $2,750.00 |  $50.00 | 10159   | Rs.      | $3,000.00 |  $300.00 | GF611
preet1s6@yahoo.com      | JRG         | 2021-08-02 17:30:00 | KIA         | 2021-08-02 21:10:00 | preet02aug  |       2 | $2,500.00 | $100.00 | 10111   | Rs.      | $2,400.00 |    $0.00 | JA617
preet1s6@yahoo.com      | CNN         | 2021-07-14 08:00:00 | IXE         | 2021-07-14 15:30:00 | preet14jul  |       2 | $3,100.00 |  $82.00 | 10112   | Rs.      | $3,100.00 |   $82.00 | OG230
(3 rows)
```

- **All the travellers who got a discount**

```
select * from flight_trip where discount>'$0';
```

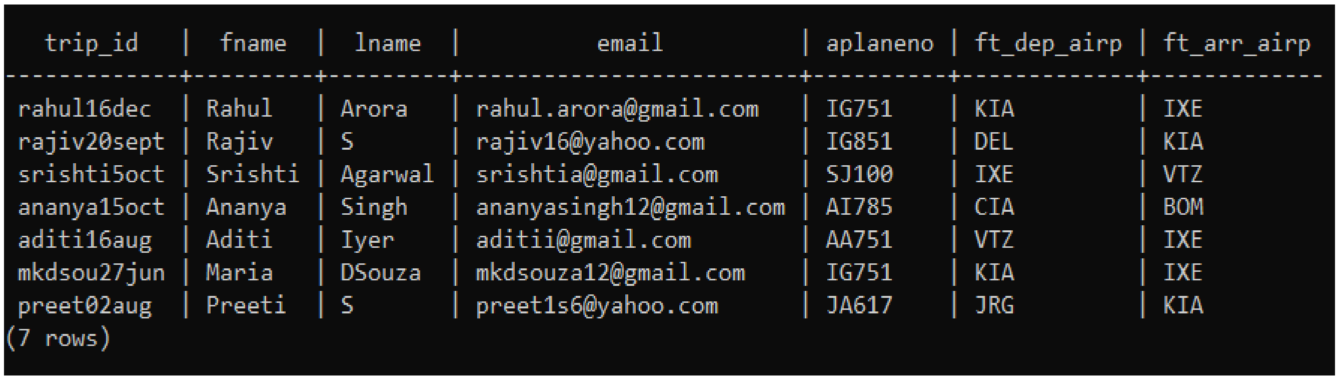


```
         email          | ft_dep_airp |     ft_dep_time     | ft_arr_airp |      arr_time       |   trip_id   | no_trav |  tot_amt  |   tax   | tran_id | currency | base_amt  | discount | aplaneno
------------------------+-------------+---------------------+-------------+---------------------+-------------+---------+-----------+---------+---------+----------+-----------+----------+---------
aditii@gmail.com        | VTZ         | 2021-08-07 21:00:00 | IXE         | 2021-08-07 21:45:00 | aditi16aug  |       1 | $2,750.00 |  $50.00 | 10159   | Rs.      | $3,000.00 |  $300.00 | AA751
rahul.arora@gmail.com   | KIA         | 2021-06-27 12:10:25 | IXE         | 2021-06-27 13:00:00 | rahul16dec  |       1 | $3,000.00 | $100.00 | 10110   | Rs.      | $3,000.00 |  $100.00 | IG751
srishtia@gmail.com      | IXE         | 2021-07-14 08:00:00 | VTZ         | 2021-07-14 10:00:00 | srishti5oct |       1 | $3,100.00 |  $82.00 | 10112   | Rs.      | $3,100.00 |   $82.00 | SJ100
aditi.hasyagar@gmail.com | TRV        | 2021-10-12 12:30:00 | IMF         | 2021-10-12 20:00:00 | adihas12oct |       3 | $2,750.00 |  $50.00 | 10159   | Rs.      | $3,000.00 |  $300.00 | GF611
mkdsouza12@gmail.com    | KIA         | 2021-06-27 12:10:25 | IXE         | 2021-06-27 13:00:00 | mkdsou27jun |       1 | $3,000.00 | $100.00 | 10110   | Rs.      | $3,000.00 |  $100.00 | IG751
preet1s6@yahoo.com      | CNN         | 2021-07-14 08:00:00 | IXE         | 2021-07-14 15:30:00 | preet14jul  |       2 | $3,100.00 |  $82.00 | 10112   | Rs.      | $3,100.00 |   $82.00 | OG230
(6 rows)
```

COMPLEX Queries

- **All trips where the user who booked it are the ppl that travelled**

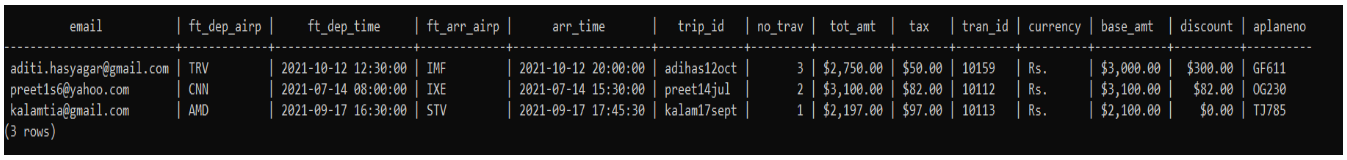
```
select t.trip_id, t.fname, t.lname, u.email, f.aplaneno, f.ft_dep_airp, f.ft_arr_airp from flight_trip F, user1 U, traveller T where U.email=F.email and F.trip_id=T.trip_id and U.fname=T.fname and U.lname=T.lname;
```



```
   trip_id    |  fname   |  lname   |           email           | aplaneno | ft_dep_airp | ft_arr_airp
-------------+---------+---------+-------------------------+----------+-------------+-------------
 rahul16dec   | Rahul    | Arora    | rahul.arora@gmail.com     | IG751    | KIA         | IXE
 rajiv20sept  | Rajiv    | S        | rajiv16@yahoo.com         | IG851    | DEL         | KIA
 srishti5oct  | Srishti  | Agarwal  | srishtia@gmail.com        | SJ100    | IXE         | VTZ
 ananya15oct  | Ananya   | Singh    | ananyasingh12@gmail.com   | AI785    | CIA         | BOM
 aditi16aug   | Aditi    | Iyer     | aditii@gmail.com          | AA751    | VTZ         | IXE
 mkdsou27jun  | Maria    | DSouza   | mkdsouza12@gmail.com      | IG751    | KIA         | IXE
 preet02aug   | Preeti   | S        | preet1s6@yahoo.com        | JA617    | JRG         | KIA
(7 rows)
```

- **All the trips where ppl that booked weren't the people that travelled**

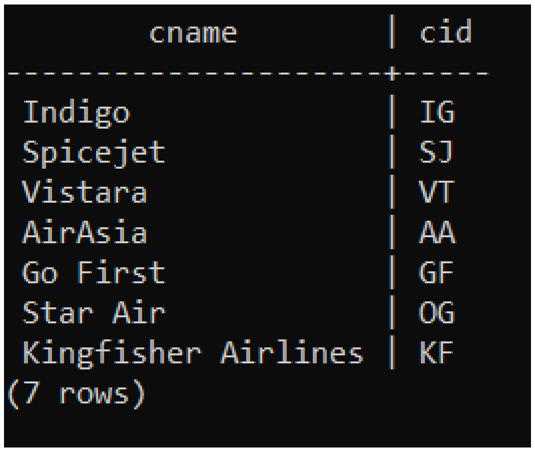
```
select * from flight_trip F where f.trip_id not in (select f.trip_id from flight_trip F, user1 U, traveller T where U.email=F.email and F.trip_id=T.trip_id and U.fname=T.fname and U.lname=T.lname);
```



```
         email          | ft_dep_airp |     ft_dep_time     | ft_arr_airp |      arr_time       |   trip_id   | no_trav |  tot_amt  |   tax   | tran_id | currency |  base_amt  | discount | aplaneno
------------------------+-------------+---------------------+-------------+---------------------+-------------+---------+-----------+---------+---------+----------+------------+----------+----------
 aditi.hasyagar@gmail.com | TRV       | 2021-10-12 12:30:00 | IMF         | 2021-10-12 20:00:00 | adihas12oct |       3 | $2,750.00 | $50.00  | 10159   | Rs.      | $3,000.00  |  $300.00 | GF611
 preet1s6@yahoo.com     | CNN         | 2021-07-14 08:00:00 | IXE         | 2021-07-14 15:30:00 | preet14jul  |       2 | $3,100.00 | $82.00  | 10112   | Rs.      | $3,100.00  |   $82.00 | OG230
 kalamtia@gmail.com     | AMD         | 2021-09-17 16:30:00 | STV         | 2021-09-17 17:45:30 | kalam17sept |       1 | $2,197.00 | $97.00  | 10113   | Rs.      | $2,100.00  |    $0.00 | TJ785
(3 rows)
```

- **All the companies that dont have flights to bangalore**

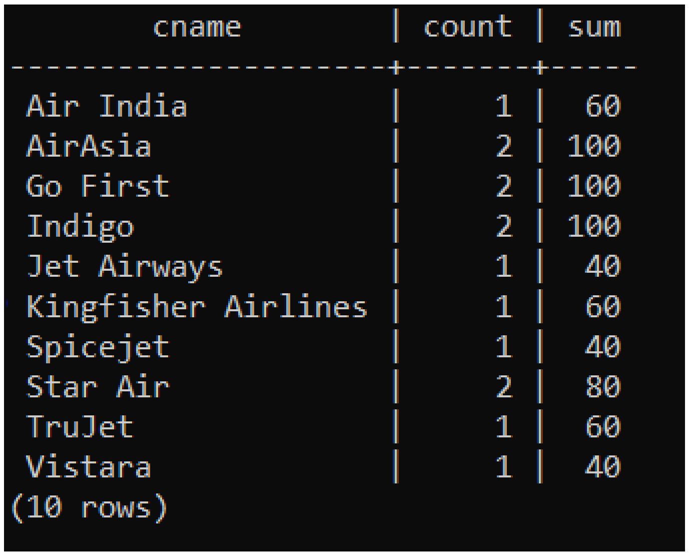
```
select * from airline_company where cid in (select cid from aeroplane where aplaneno in (select aplaneno from stops_at where aplaneno not in (select aplaneno from stops_at where acode='KIA')));
```



```
        cname        | cid
---------------------+-----
 Indigo              | IG
 Spicejet            | SJ
 Vistara             | VT
 AirAsia             | AA
 Go First            | GF
 Star Air            | OG
 Kingfisher Airlines | KF
(7 rows)
```

- **A count of the flights chartered by each company and the total seats**

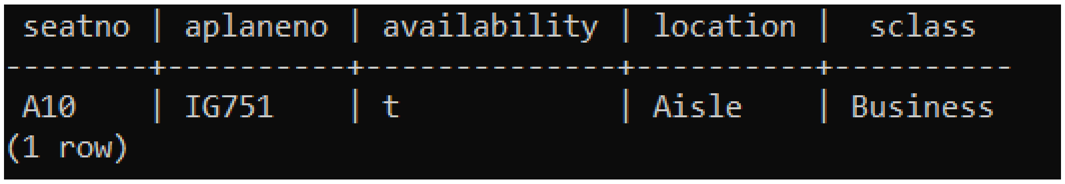
```
select cname, count(distinct aplaneno), sum(tot_seats) from
airline_company natural join aeroplane group by cname;
```



```
        cname         | count | sum
----------------------+-------+-----
 Air India            |     1 |  60
 AirAsia              |     2 | 100
 Go First             |     2 | 100
 Indigo               |     2 | 100
 Jet Airways          |     1 |  40
 Kingfisher Airlines  |     1 |  60
 Spicejet             |     1 |  40
 Star Air             |     2 |  80
 TruJet               |     1 |  60
 Vistara              |     1 |  40
(10 rows)
```

- **Details of all the seats that on IG751 that have not been reserved**

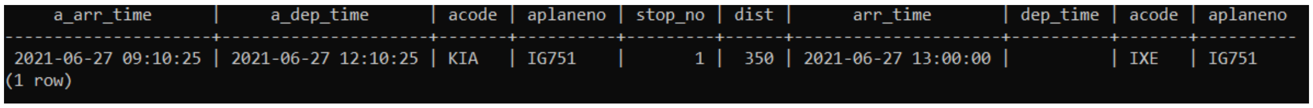
```
select * from seat where seatno not in (select seatno from
reserved where aplaneno='IG751') and aplaneno='IG751';
```



```
 seatno | aplaneno | availability | location |  sclass
--------+----------+--------------+----------+----------
 A10    | IG751    | t            | Aisle    | Business
(1 row)
```

- **Details of all the flights flying fromBangalore to Bombay without layover on the 27th of Jun**

```
select * from arrives_at a, stops_at s where
a.aplaneno=s.aplaneno and a.acode='KIA' and s.acode='IXE' and
stop_no=1 and date(a_dep_time)='2021-06-27';
```



```
      a_arr_time      |      a_dep_time      | acode | aplaneno | stop_no | dist |       arr_time       | dep_time | acode | aplaneno
---------------------+---------------------+-------+----------+---------+------+---------------------+----------+-------+----------
 2021-06-27 09:10:25 | 2021-06-27 12:10:25 | KIA   | IG751    |       1 |  350 | 2021-06-27 13:00:00 |          | IXE   | IG751
(1 row)
```

QUERY EXECUTION PLAN

Some terms, explained:

Seq Scan

The Seq Scan operation scans the entire relation (table) as stored on disk (like TABLE ACCESS FULL).

Nested Loops

Joins two tables by fetching the result from one table and querying the other table for each row from the first.

Hash Join / Hash

The hash join loads the candidate records from one side of the join into a hash table (marked with Hash in the plan) which is then probed for each record from the other side of the join.

Sort / Sort Key

Sorts the set on the columns mentioned in Sort Key. The Sort operation needs large amounts of memory to materialize the intermediate result (not pipelined).

GroupAggregate

Aggregates a presorted set according to the group by clause. This operation does not buffer large amounts of data (pipelined).

HashAggregate

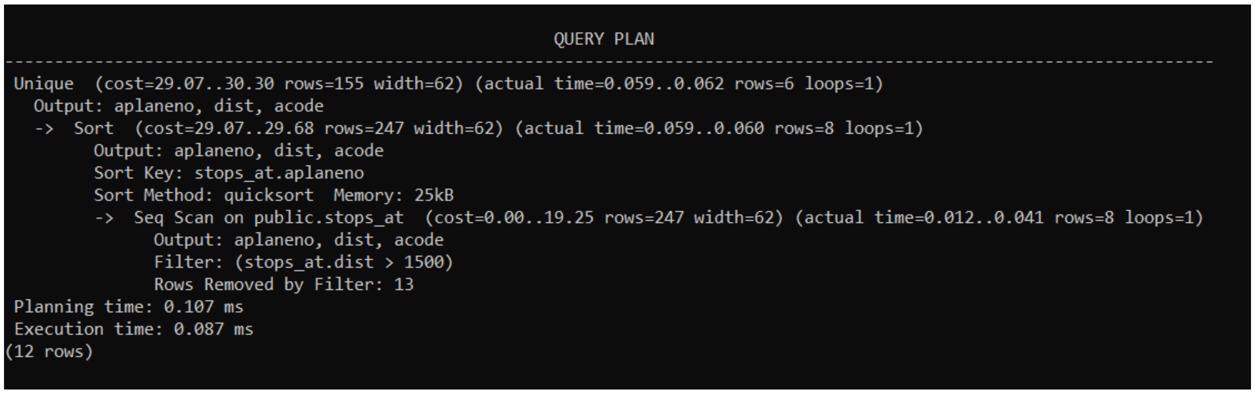
Uses a temporary hash table to group records. The HashAggregate operation does not require a presorted data set, instead it uses large amounts of memory to materialize the intermediate result (not pipelined). The output is not ordered in any meaningful way

Limit

Aborts the underlying operations when the desired number of rows has been fetched.

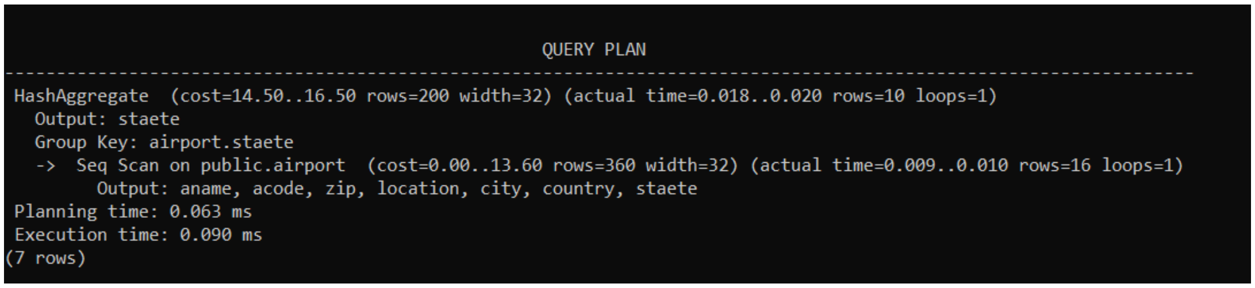
Simple Queries

```
explain analyze verbose select distinct on (aplaneno) aplaneno, dist, acode from stops_at where dist>1500;
```



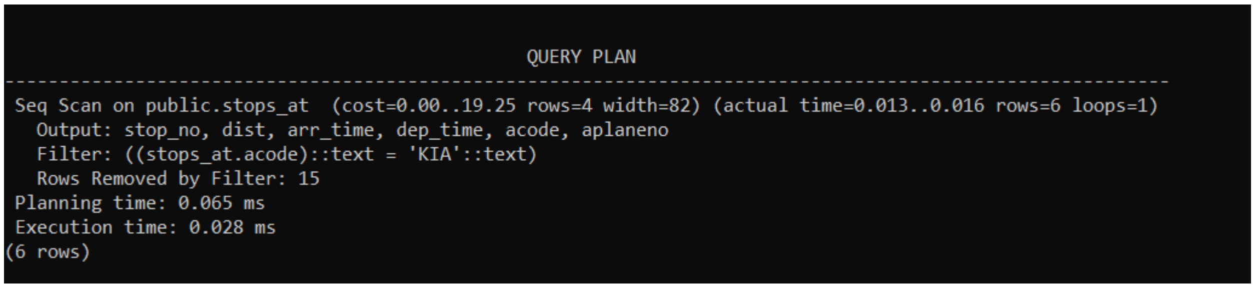
```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=29.07..30.30 rows=155 width=62) (actual time=0.059..0.062 rows=6 loops=1)
   Output: aplaneno, dist, acode
   ->  Sort  (cost=29.07..29.68 rows=247 width=62) (actual time=0.059..0.060 rows=8 loops=1)
         Output: aplaneno, dist, acode
         Sort Key: stops_at.aplaneno
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on public.stops_at  (cost=0.00..19.25 rows=247 width=62) (actual time=0.012..0.041 rows=8 loops=1)
               Output: aplaneno, dist, acode
               Filter: (stops_at.dist > 1500)
               Rows Removed by Filter: 13
 Planning time: 0.107 ms
 Execution time: 0.087 ms
(12 rows)
```

```
explain analyze verbose select staete from airport group by staete;
```



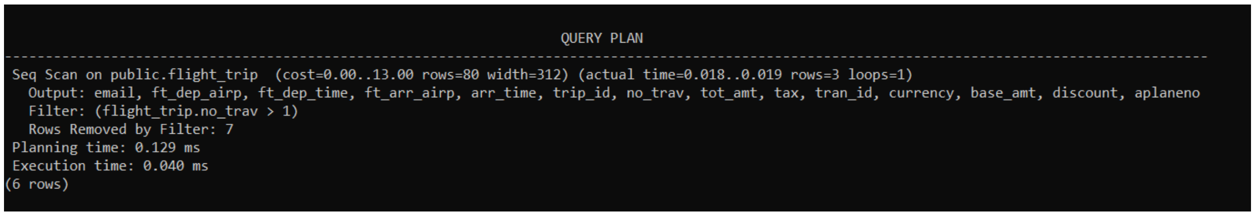
```
                                                    QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=14.50..16.50 rows=200 width=32) (actual time=0.018..0.020 rows=10 loops=1)
   Output: staete
   Group Key: airport.staete
   ->  Seq Scan on public.airport  (cost=0.00..13.60 rows=360 width=32) (actual time=0.009..0.010 rows=16 loops=1)
         Output: aname, acode, zip, location, city, country, staete
 Planning time: 0.063 ms
 Execution time: 0.090 ms
(7 rows)
```

```
explain analyze verbose select * from stops_at where acode='KIA';
```



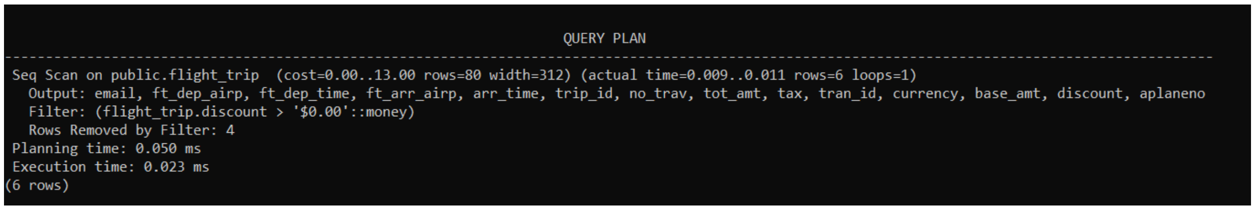
```
                                              QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Seq Scan on public.stops_at  (cost=0.00..19.25 rows=4 width=82) (actual time=0.013..0.016 rows=6 loops=1)
   Output: stop_no, dist, arr_time, dep_time, acode, aplaneno
   Filter: ((stops_at.acode)::text = 'KIA'::text)
   Rows Removed by Filter: 15
 Planning time: 0.065 ms
 Execution time: 0.028 ms
(6 rows)
```

explain analyze verbose select * from flight_trip where no_trav>1;



```
                                                            QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on public.flight_trip  (cost=0.00..13.00 rows=80 width=312) (actual time=0.018..0.019 rows=3 loops=1)
   Output: email, ft_dep_airp, ft_dep_time, ft_arr_airp, arr_time, trip_id, no_trav, tot_amt, tax, tran_id, currency, base_amt, discount, aplaneno
   Filter: (flight_trip.no_trav > 1)
   Rows Removed by Filter: 7
 Planning time: 0.129 ms
 Execution time: 0.040 ms
(6 rows)
```

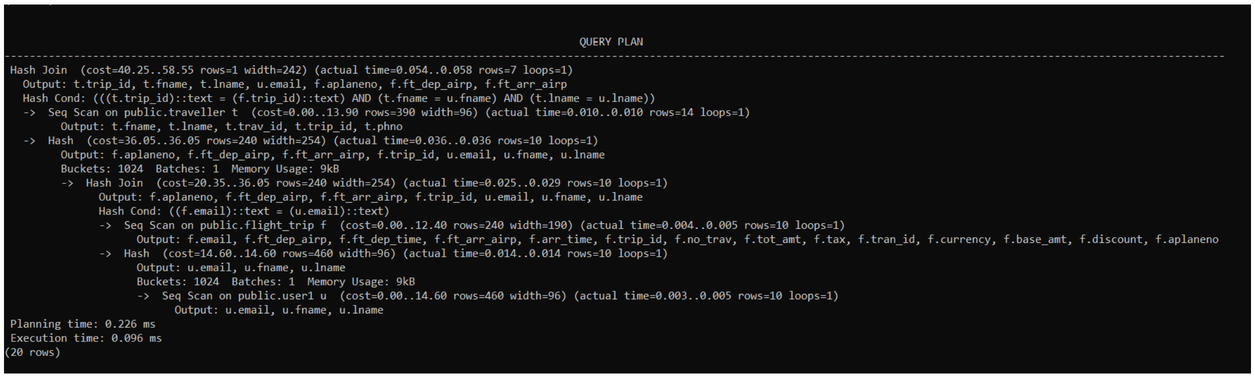
explain analyze verbose select * from flight_trip where discount>'$0';



```
                                                            QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on public.flight_trip  (cost=0.00..13.00 rows=80 width=312) (actual time=0.009..0.011 rows=6 loops=1)
   Output: email, ft_dep_airp, ft_dep_time, ft_arr_airp, arr_time, trip_id, no_trav, tot_amt, tax, tran_id, currency, base_amt, discount, aplaneno
   Filter: (flight_trip.discount > '$0.00'::money)
   Rows Removed by Filter: 4
 Planning time: 0.050 ms
 Execution time: 0.023 ms
(6 rows)
```

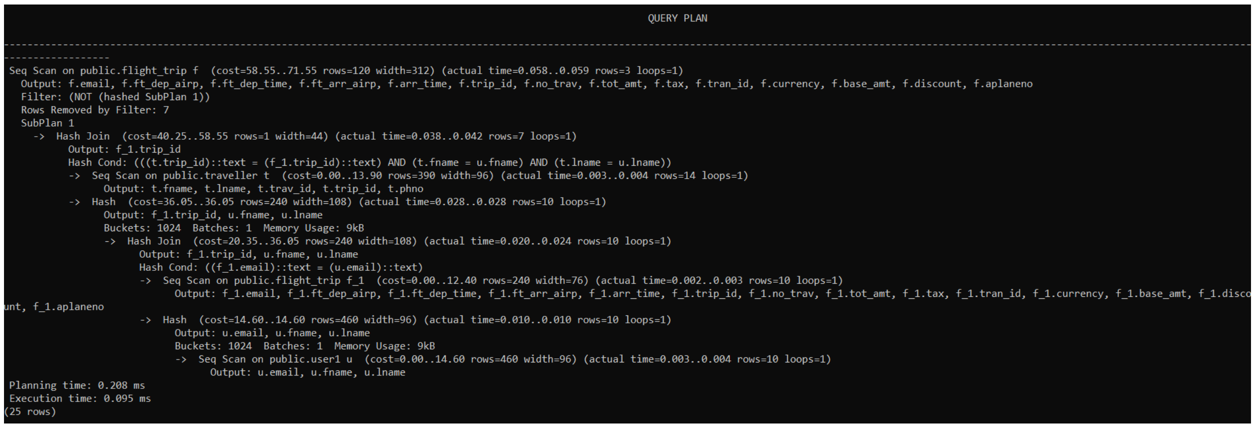
Complex Queries

explain analyze verbose select t.trip_id, t.fname, t.lname, u.email, f.aplaneno, f.ft_dep_airp, f.ft_arr_airp from flight_trip F, user1 U, traveller T where U.email=F.email and F.trip_id=T.trip_id and U.fname=T.fname and U.lname=T.lname;



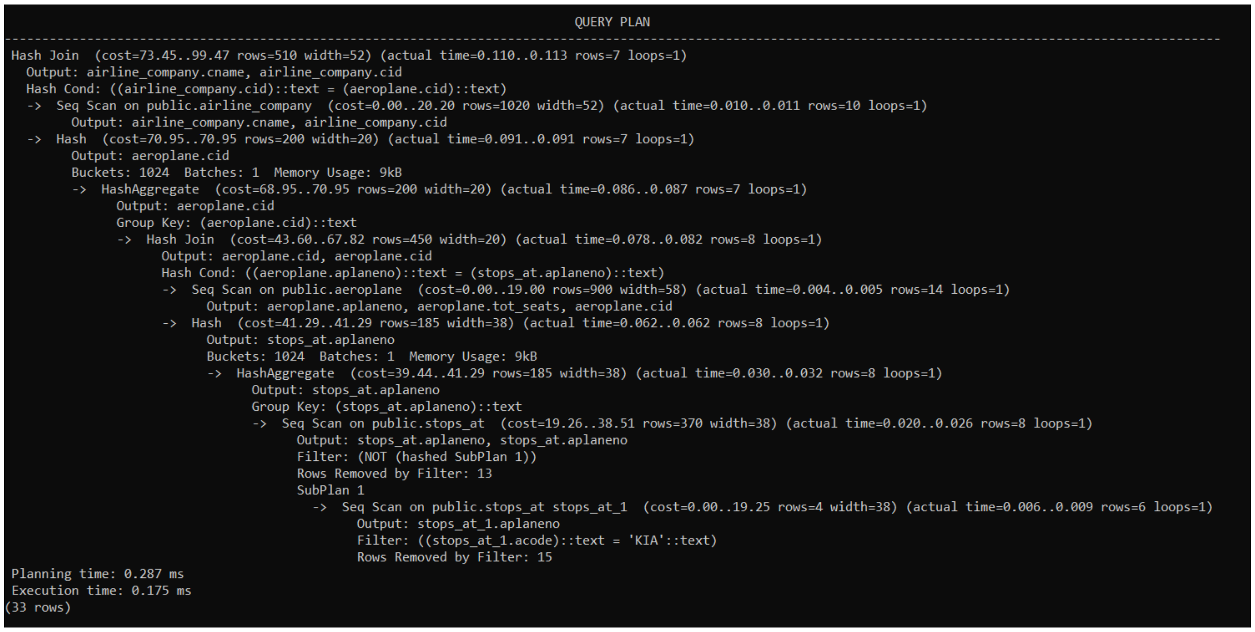
```
                                                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=40.25..58.55 rows=1 width=242) (actual time=0.054..0.058 rows=7 loops=1)
   Output: t.trip_id, t.fname, t.lname, u.email, f.aplaneno, f.ft_dep_airp, f.ft_arr_airp
   Hash Cond: (((t.trip_id)::text = (f.trip_id)::text) AND (t.fname = u.fname) AND (t.lname = u.lname))
   ->  Seq Scan on public.traveller t  (cost=0.00..13.90 rows=390 width=96) (actual time=0.010..0.010 rows=14 loops=1)
         Output: t.fname, t.lname, t.trav_id, t.trip_id, t.phno
   ->  Hash  (cost=36.05..36.05 rows=240 width=254) (actual time=0.036..0.036 rows=10 loops=1)
         Output: f.aplaneno, f.ft_dep_airp, f.ft_arr_airp, f.trip_id, u.email, u.fname, u.lname
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Hash Join  (cost=20.35..36.05 rows=240 width=254) (actual time=0.025..0.029 rows=10 loops=1)
               Output: f.aplaneno, f.ft_dep_airp, f.ft_arr_airp, f.trip_id, u.email, u.fname, u.lname
               Hash Cond: ((f.email)::text = (u.email)::text)
               ->  Seq Scan on public.flight_trip f  (cost=0.00..12.40 rows=240 width=190) (actual time=0.004..0.005 rows=10 loops=1)
                     Output: f.email, f.ft_dep_airp, f.ft_dep_time, f.ft_arr_airp, f.arr_time, f.trip_id, f.no_trav, f.tot_amt, f.tax, f.tran_id, f.currency, f.base_amt, f.discount, f.aplaneno
               ->  Hash  (cost=14.60..14.60 rows=460 width=96) (actual time=0.014..0.014 rows=10 loops=1)
                     Output: u.email, u.fname, u.lname
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on public.user1 u  (cost=0.00..14.60 rows=460 width=96) (actual time=0.003..0.005 rows=10 loops=1)
                           Output: u.email, u.fname, u.lname
 Planning time: 0.226 ms
 Execution time: 0.096 ms
(20 rows)
```

explain analyze verbose select * from flight_trip F where f.trip_id not in (select f.trip_id from flight_trip F, user1 U, traveller T where U.email=F.email and F.trip_id=T.trip_id and U.fname=T.fname and U.lname=T.lname);



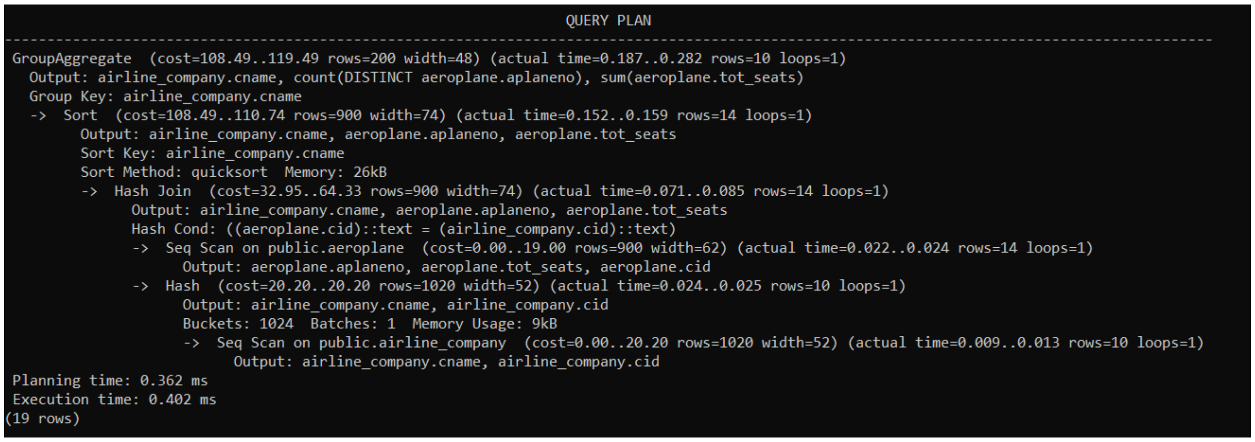
```
                                                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------
 Seq Scan on public.flight_trip f  (cost=58.55..71.55 rows=120 width=312) (actual time=0.058..0.059 rows=3 loops=1)
   Output: f.email, f.ft_dep_airp, f.ft_dep_time, f.ft_arr_airp, f.arr_time, f.trip_id, f.no_trav, f.tot_amt, f.tax, f.tran_id, f.currency, f.base_amt, f.discount, f.aplaneno
   Filter: (NOT (hashed SubPlan 1))
   Rows Removed by Filter: 7
   SubPlan 1
     ->  Hash Join  (cost=40.25..58.55 rows=1 width=44) (actual time=0.038..0.042 rows=7 loops=1)
           Output: f_1.trip_id
           Hash Cond: (((t.trip_id)::text = (f_1.trip_id)::text) AND (t.fname = u.fname) AND (t.lname = u.lname))
           ->  Seq Scan on public.traveller t  (cost=0.00..13.90 rows=390 width=96) (actual time=0.003..0.004 rows=14 loops=1)
                 Output: t.fname, t.lname, t.trav_id, t.trip_id, t.phno
           ->  Hash  (cost=36.05..36.05 rows=240 width=108) (actual time=0.028..0.028 rows=10 loops=1)
                 Output: f_1.trip_id, u.fname, u.lname
                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
                 ->  Hash Join  (cost=20.35..36.05 rows=240 width=108) (actual time=0.020..0.024 rows=10 loops=1)
                       Output: f_1.trip_id, u.fname, u.lname
                       Hash Cond: ((f_1.email)::text = (u.email)::text)
                       ->  Seq Scan on public.flight_trip f_1  (cost=0.00..12.40 rows=240 width=76) (actual time=0.002..0.003 rows=10 loops=1)
                             Output: f_1.email, f_1.ft_dep_airp, f_1.ft_dep_time, f_1.ft_arr_airp, f_1.arr_time, f_1.trip_id, f_1.no_trav, f_1.tot_amt, f_1.tax, f_1.tran_id, f_1.currency, f_1.base_amt, f_1.disco
unt, f_1.aplaneno
                       ->  Hash  (cost=14.60..14.60 rows=460 width=96) (actual time=0.010..0.010 rows=10 loops=1)
                             Output: u.email, u.fname, u.lname
                             Buckets: 1024  Batches: 1  Memory Usage: 9kB
                             ->  Seq Scan on public.user1 u  (cost=0.00..14.60 rows=460 width=96) (actual time=0.003..0.004 rows=10 loops=1)
                                   Output: u.email, u.fname, u.lname
 Planning time: 0.208 ms
 Execution time: 0.095 ms
(25 rows)
```

explain analyze verbose select * from airline_company where cid in (select cid from aeroplane where aplaneno in (select aplaneno from stops_at where aplaneno not in (select aplaneno from stops_at where acode='KIA')));



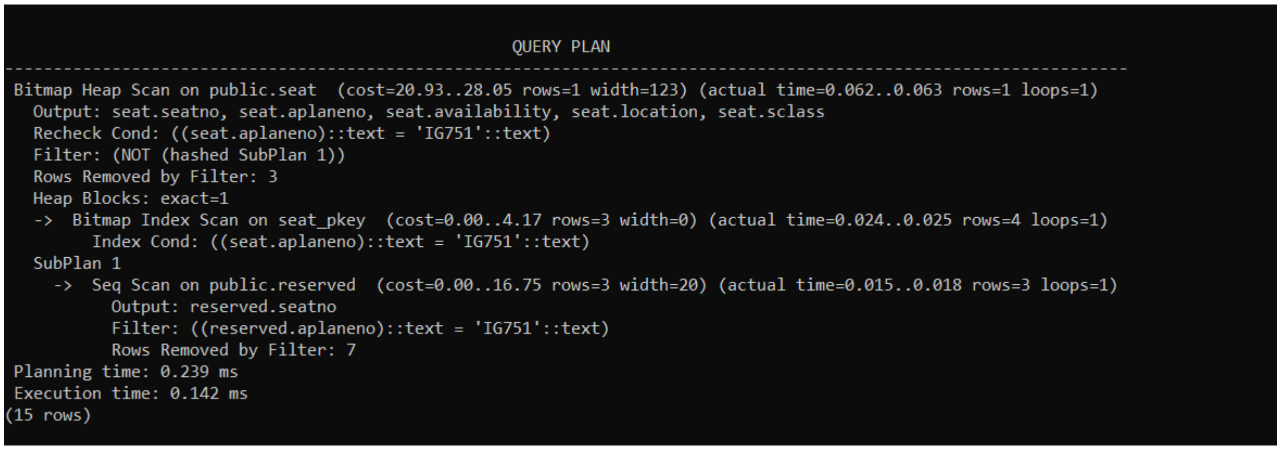
```
                                                                                QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=73.45..99.47 rows=510 width=52) (actual time=0.110..0.113 rows=7 loops=1)
   Output: airline_company.cname, airline_company.cid
   Hash Cond: ((airline_company.cid)::text = (aeroplane.cid)::text)
   ->  Seq Scan on public.airline_company  (cost=0.00..20.20 rows=1020 width=52) (actual time=0.010..0.011 rows=10 loops=1)
         Output: airline_company.cname, airline_company.cid
   ->  Hash  (cost=70.95..70.95 rows=200 width=20) (actual time=0.091..0.091 rows=7 loops=1)
         Output: aeroplane.cid
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  HashAggregate  (cost=68.95..70.95 rows=200 width=20) (actual time=0.086..0.087 rows=7 loops=1)
               Output: aeroplane.cid
               Group Key: (aeroplane.cid)::text
               ->  Hash Join  (cost=43.60..67.82 rows=450 width=20) (actual time=0.078..0.082 rows=8 loops=1)
                     Output: aeroplane.cid, aeroplane.cid
                     Hash Cond: ((aeroplane.aplaneno)::text = (stops_at.aplaneno)::text)
                     ->  Seq Scan on public.aeroplane  (cost=0.00..19.00 rows=900 width=58) (actual time=0.004..0.005 rows=14 loops=1)
                           Output: aeroplane.aplaneno, aeroplane.tot_seats, aeroplane.cid
                     ->  Hash  (cost=41.29..41.29 rows=185 width=38) (actual time=0.062..0.062 rows=8 loops=1)
                           Output: stops_at.aplaneno
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  HashAggregate  (cost=39.44..41.29 rows=185 width=38) (actual time=0.030..0.032 rows=8 loops=1)
                                 Output: stops_at.aplaneno
                                 Group Key: (stops_at.aplaneno)::text
                                 ->  Seq Scan on public.stops_at  (cost=19.26..38.51 rows=370 width=38) (actual time=0.020..0.026 rows=8 loops=1)
                                       Output: stops_at.aplaneno, stops_at.aplaneno
                                       Filter: (NOT (hashed SubPlan 1))
                                       Rows Removed by Filter: 13
                                       SubPlan 1
                                         ->  Seq Scan on public.stops_at stops_at_1  (cost=0.00..19.25 rows=4 width=38) (actual time=0.006..0.009 rows=6 loops=1)
                                               Output: stops_at_1.aplaneno
                                               Filter: ((stops_at_1.acode)::text = 'KIA'::text)
                                               Rows Removed by Filter: 15
 Planning time: 0.287 ms
 Execution time: 0.175 ms
(33 rows)
```

explain analyze verbose select cname, count(distinct aplaneno), sum(tot_seats) from airline_company natural join aeroplane group by cname;



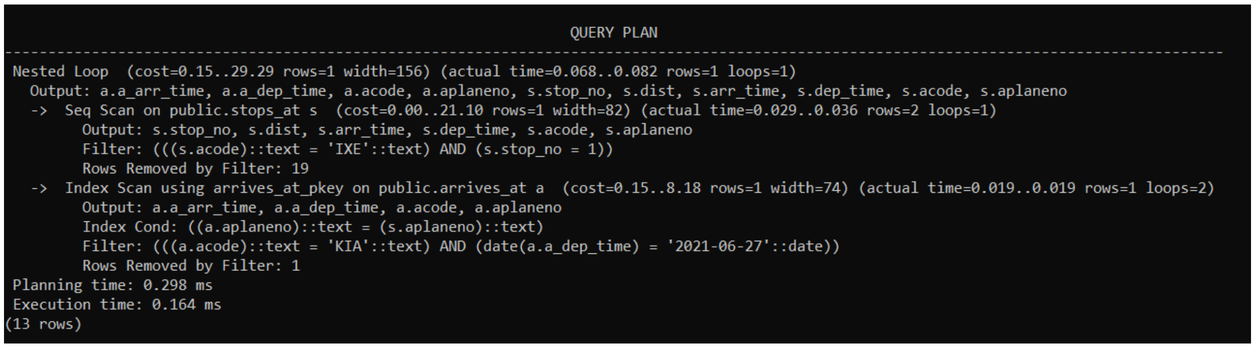
```
                                                                                QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=108.49..119.49 rows=200 width=48) (actual time=0.187..0.282 rows=10 loops=1)
   Output: airline_company.cname, count(DISTINCT aeroplane.aplaneno), sum(aeroplane.tot_seats)
   Group Key: airline_company.cname
   ->  Sort  (cost=108.49..110.74 rows=900 width=74) (actual time=0.152..0.159 rows=14 loops=1)
         Output: airline_company.cname, aeroplane.aplaneno, aeroplane.tot_seats
         Sort Key: airline_company.cname
         Sort Method: quicksort  Memory: 26kB
         ->  Hash Join  (cost=32.95..64.33 rows=900 width=74) (actual time=0.071..0.085 rows=14 loops=1)
               Output: airline_company.cname, aeroplane.aplaneno, aeroplane.tot_seats
               Hash Cond: ((aeroplane.cid)::text = (airline_company.cid)::text)
               ->  Seq Scan on public.aeroplane  (cost=0.00..19.00 rows=900 width=62) (actual time=0.022..0.024 rows=14 loops=1)
                     Output: aeroplane.aplaneno, aeroplane.tot_seats, aeroplane.cid
               ->  Hash  (cost=20.20..20.20 rows=1020 width=52) (actual time=0.024..0.025 rows=10 loops=1)
                     Output: airline_company.cname, airline_company.cid
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on public.airline_company  (cost=0.00..20.20 rows=1020 width=52) (actual time=0.009..0.013 rows=10 loops=1)
                           Output: airline_company.cname, airline_company.cid
 Planning time: 0.362 ms
 Execution time: 0.402 ms
(19 rows)
```

explain analyze verbose select * from seat where seatno not in (select seatno from reserved where aplaneno='IG751') and aplaneno='IG751';



```
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on public.seat  (cost=20.93..28.05 rows=1 width=123) (actual time=0.062..0.063 rows=1 loops=1)
   Output: seat.seatno, seat.aplaneno, seat.availability, seat.location, seat.sclass
   Recheck Cond: ((seat.aplaneno)::text = 'IG751'::text)
   Filter: (NOT (hashed SubPlan 1))
   Rows Removed by Filter: 3
   Heap Blocks: exact=1
   ->  Bitmap Index Scan on seat_pkey  (cost=0.00..4.17 rows=3 width=0) (actual time=0.024..0.025 rows=4 loops=1)
         Index Cond: ((seat.aplaneno)::text = 'IG751'::text)
   SubPlan 1
     ->  Seq Scan on public.reserved  (cost=0.00..16.75 rows=3 width=20) (actual time=0.015..0.018 rows=3 loops=1)
           Output: reserved.seatno
           Filter: ((reserved.aplaneno)::text = 'IG751'::text)
           Rows Removed by Filter: 7
 Planning time: 0.239 ms
 Execution time: 0.142 ms
(15 rows)
```

```
explain analyze verbose select * from arrives_at a, stops_at s
where a.aplaneno=s.aplaneno and a.acode='KIA' and s.acode='IXE'
and stop_no=1 and date(a_dep_time)='2021-06-27';
```



```
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.15..29.29 rows=1 width=156) (actual time=0.068..0.082 rows=1 loops=1)
   Output: a.a_arr_time, a.a_dep_time, a.acode, a.aplaneno, s.stop_no, s.dist, s.arr_time, s.dep_time, s.acode, s.aplaneno
   ->  Seq Scan on public.stops_at s  (cost=0.00..21.10 rows=1 width=82) (actual time=0.029..0.036 rows=2 loops=1)
         Output: s.stop_no, s.dist, s.arr_time, s.dep_time, s.acode, s.aplaneno
         Filter: (((s.acode)::text = 'IXE'::text) AND (s.stop_no = 1))
         Rows Removed by Filter: 19
   ->  Index Scan using arrives_at_pkey on public.arrives_at a  (cost=0.15..8.18 rows=1 width=74) (actual time=0.019..0.019 rows=1 loops=2)
         Output: a.a_arr_time, a.a_dep_time, a.acode, a.aplaneno
         Index Cond: ((a.aplaneno)::text = (s.aplaneno)::text)
         Filter: (((a.acode)::text = 'KIA'::text) AND (date(a.a_dep_time) = '2021-06-27'::date))
         Rows Removed by Filter: 1
 Planning time: 0.298 ms
 Execution time: 0.164 ms
(13 rows)
```

Creating Multiple Users

The following commands create 5 different users with 5 different types of privileges.

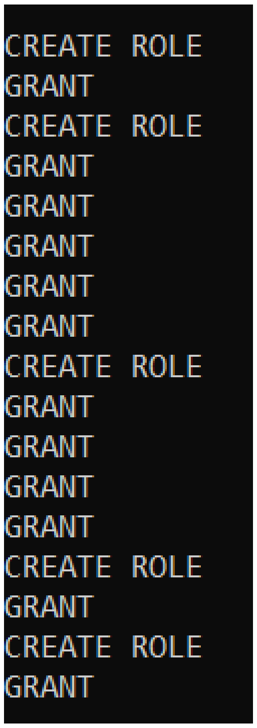
The users created are:

- Admin: This role has all the privileges on the database airres
- Tester: This user inspects the records in the relations stops_at, arrives_at, flight trip, reserved and seat. T

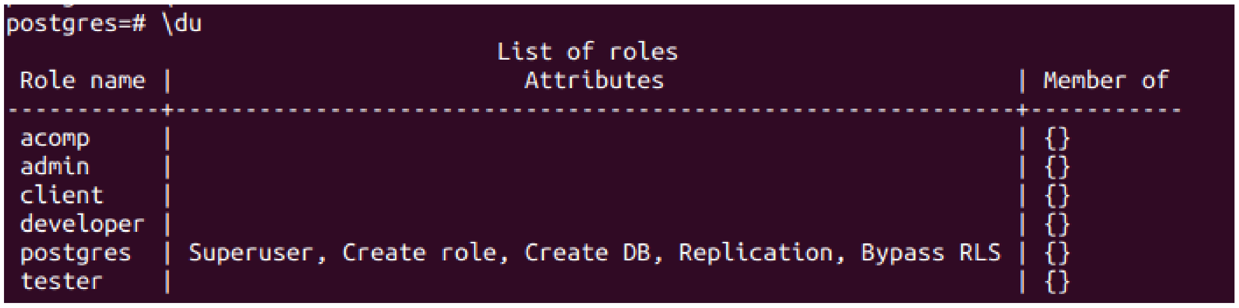
- Developer: This role can change data in all tables and create the database
- Client: This user can only retrieve and read information.
- AComp: This role was created for employees from the airline company, in case they wish to add, delete, append or create flight details.

```
create user Admin with password '12345';

grant all privileges on database airres to Admin;

create user Tester with password '12345';

grant select, update, insert, delete on table stops_at to Tester;

grant select, update, insert, delete on table arrives_at to
Tester;

grant select, update, insert, delete on table flight_trip to
Tester;

grant select, update, insert, delete on table reserved to Tester;

grant select, update, insert, delete on table seat to Tester;

create user Developer with password '12345';

GRANT SELECT,update, insert, delete ON ALL TABLES IN SCHEMA
public TO Developer;

grant create on database airres to Developer;

grant references on all tables in schema public to Developer;

create user Client with password '12345';

grant select on all tables in schema public to Client;

create user AComp with password '12345';
```

```
grant all privileges on table aeroplane, arrives_at, stops_at,
seat to Acomp;
```

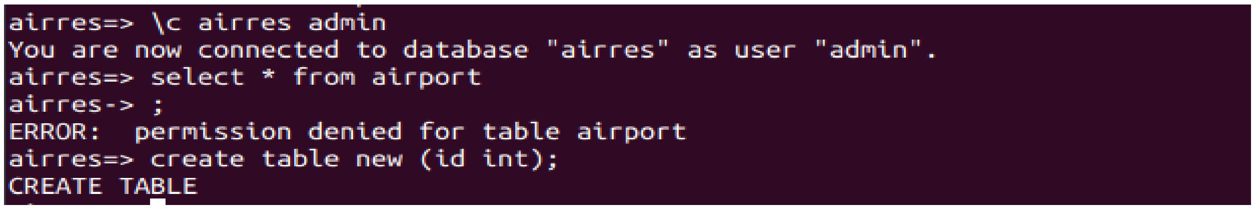


```
CREATE ROLE
GRANT
CREATE ROLE
GRANT
GRANT
GRANT
GRANT
GRANT
CREATE ROLE
GRANT
GRANT
GRANT
GRANT
CREATE ROLE
GRANT
CREATE ROLE
GRANT
```

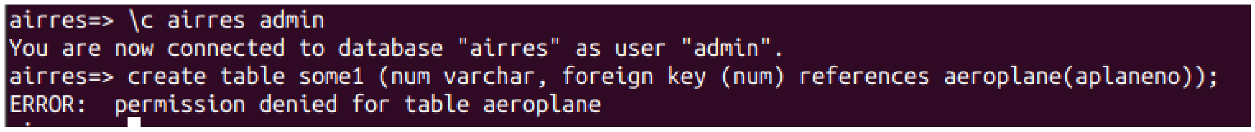


```
postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 acomp     |                                                            | {}
 admin     |                                                            | {}
 client    |                                                            | {}
 developer |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 tester    |                                                            | {}
```

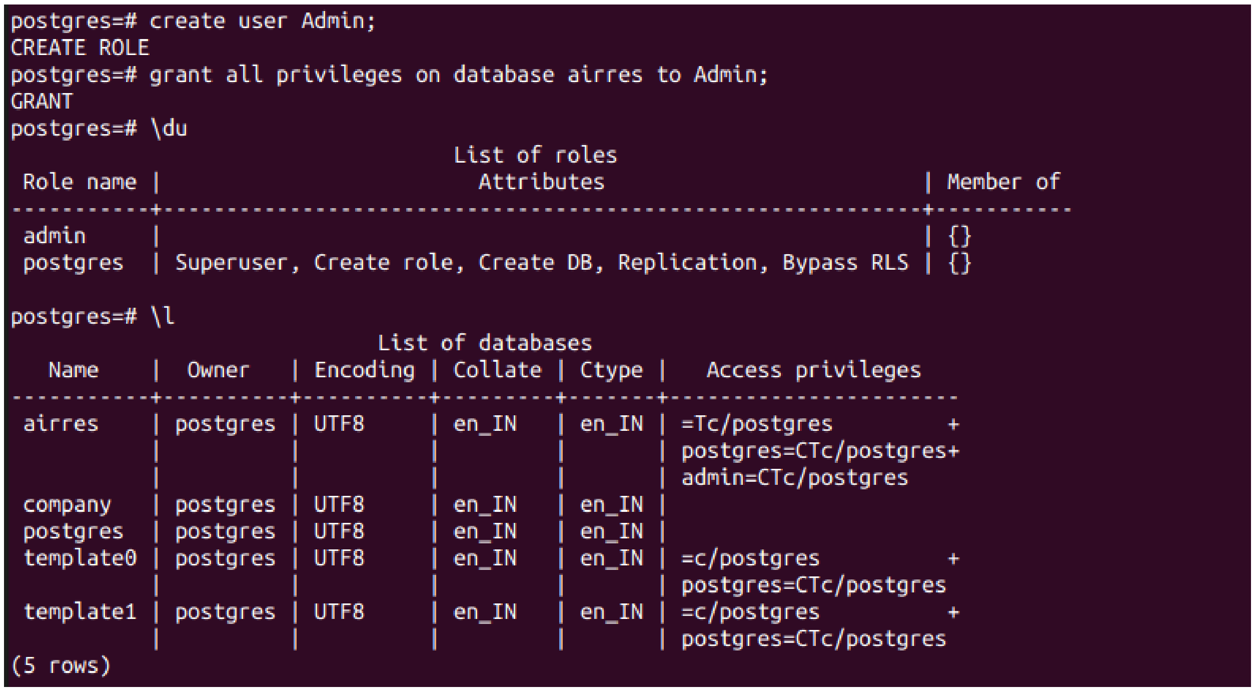
Admin:



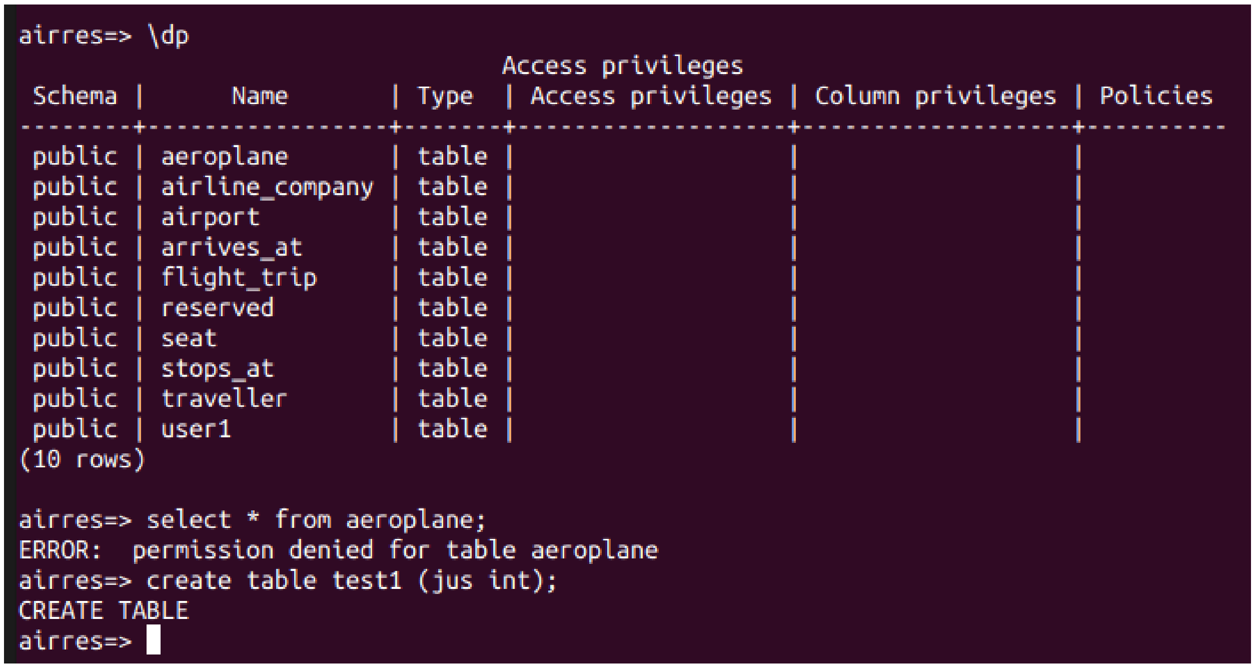
```
airres=> \c airres admin
You are now connected to database "airres" as user "admin".
airres=> select * from airport
airres-> ;
ERROR:  permission denied for table airport
airres=> create table new (id int);
CREATE TABLE
```



```
airres=> \c airres admin
You are now connected to database "airres" as user "admin".
airres=> create table some1 (num varchar, foreign key (num) references aeroplane(aplaneno));
ERROR:  permission denied for table aeroplane
```

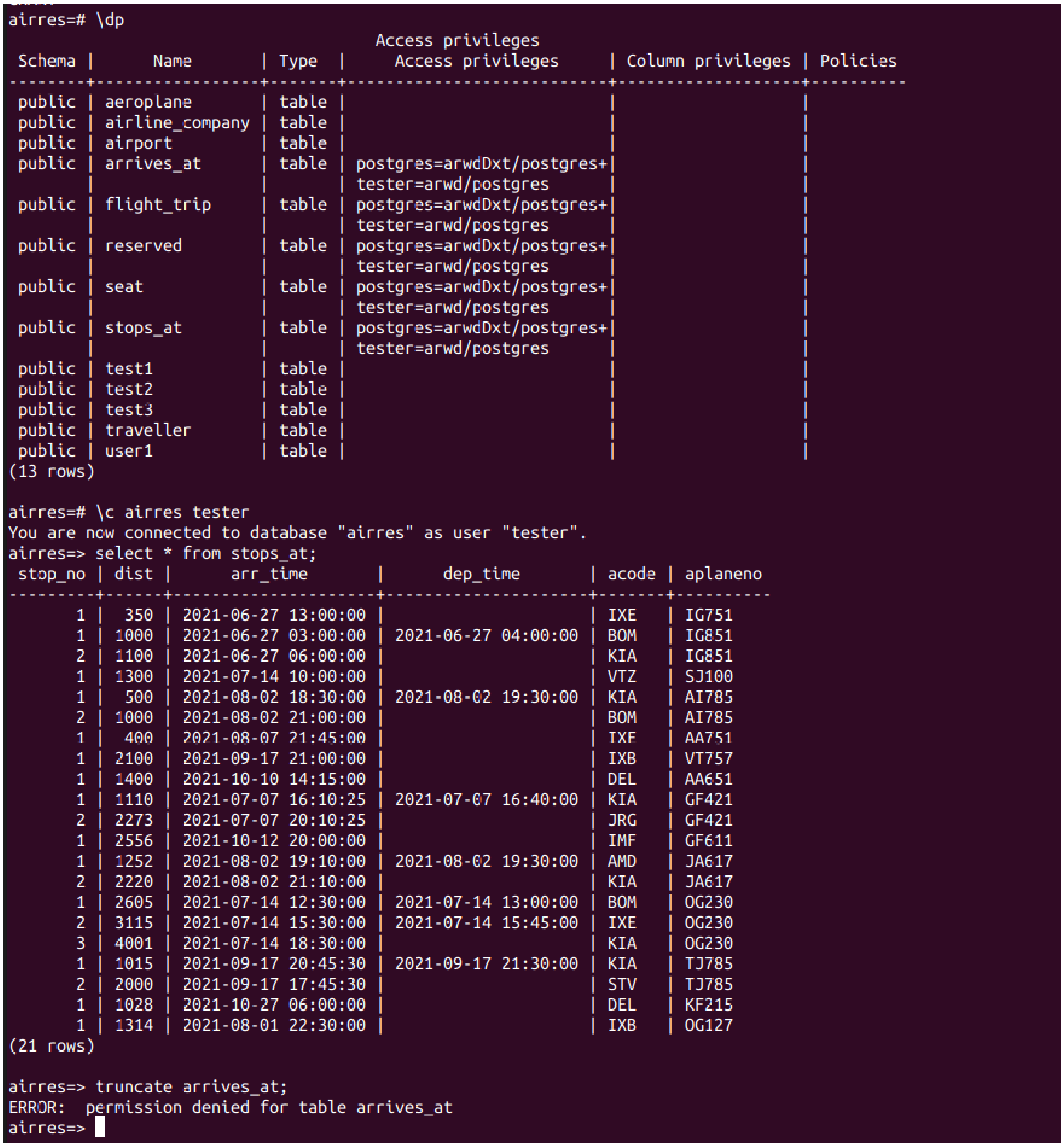


```
postgres=# create user Admin;
CREATE ROLE
postgres=# grant all privileges on database airres to Admin;
GRANT
postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 admin     |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

postgres=# \l
                             List of databases
   Name    |  Owner   | Encoding | Collate | Ctype |   Access privileges
-----------+----------+----------+---------+-------+-----------------------
 airres    | postgres | UTF8     | en_IN   | en_IN | =Tc/postgres         +
           |          |          |         |       | postgres=CTc/postgres+
           |          |          |         |       | admin=CTc/postgres
 company   | postgres | UTF8     | en_IN   | en_IN |
 postgres  | postgres | UTF8     | en_IN   | en_IN |
 template0 | postgres | UTF8     | en_IN   | en_IN | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_IN   | en_IN | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
(5 rows)
```



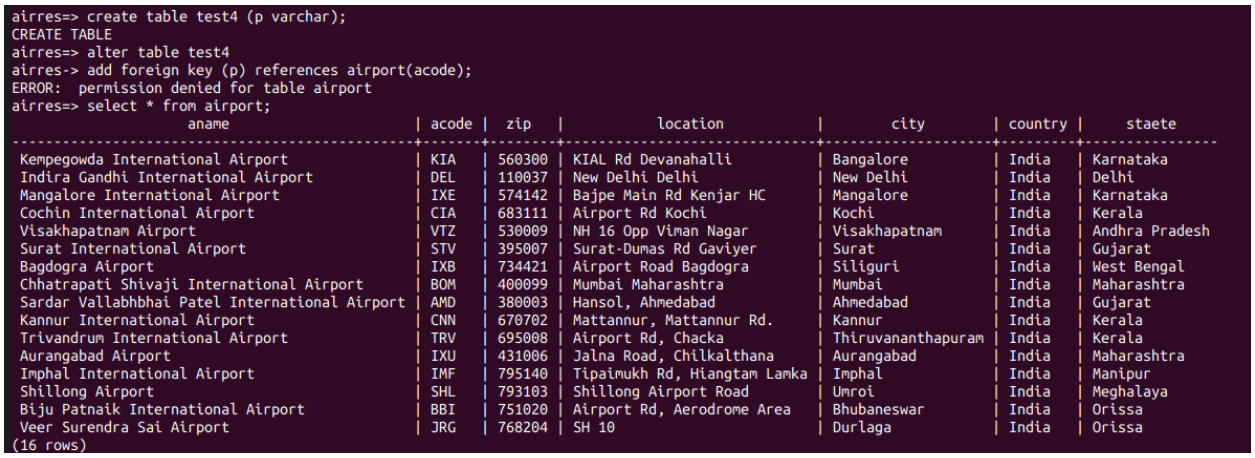
```
airres=> \dp
                                  Access privileges
 Schema |      Name       | Type  | Access privileges | Column privileges | Policies
--------+-----------------+-------+-------------------+-------------------+----------
 public | aeroplane       | table |                   |                   |
 public | airline_company | table |                   |                   |
 public | airport         | table |                   |                   |
 public | arrives_at      | table |                   |                   |
 public | flight_trip     | table |                   |                   |
 public | reserved        | table |                   |                   |
 public | seat            | table |                   |                   |
 public | stops_at        | table |                   |                   |
 public | traveller       | table |                   |                   |
 public | user1           | table |                   |                   |
(10 rows)

airres=> select * from aeroplane;
ERROR:  permission denied for table aeroplane
airres=> create table test1 (jus int);
CREATE TABLE
airres=> 
```

Tester:

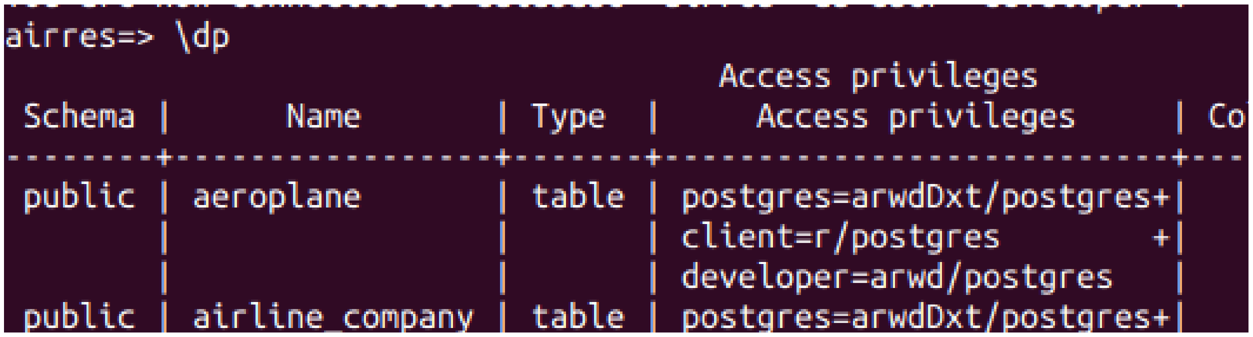


```
airres=# \dp
                                        Access privileges
 Schema |      Name       | Type  |     Access privileges      | Column privileges | Policies
--------+-----------------+-------+----------------------------+-------------------+----------
 public | aeroplane       | table |                            |                   |
 public | airline_company | table |                            |                   |
 public | airport         | table |                            |                   |
 public | arrives_at      | table | postgres=arwdDxt/postgres+|                   |
        |                 |       | tester=arwd/postgres       |                   |
 public | flight_trip     | table | postgres=arwdDxt/postgres+|                   |
        |                 |       | tester=arwd/postgres       |                   |
 public | reserved        | table | postgres=arwdDxt/postgres+|                   |
        |                 |       | tester=arwd/postgres       |                   |
 public | seat            | table | postgres=arwdDxt/postgres+|                   |
        |                 |       | tester=arwd/postgres       |                   |
 public | stops_at        | table | postgres=arwdDxt/postgres+|                   |
        |                 |       | tester=arwd/postgres       |                   |
 public | test1           | table |                            |                   |
 public | test2           | table |                            |                   |
 public | test3           | table |                            |                   |
 public | traveller       | table |                            |                   |
 public | user1           | table |                            |                   |
(13 rows)

airres=# \c airres tester
You are now connected to database "airres" as user "tester".
airres=> select * from stops_at;
 stop_no | dist |      arr_time       |      dep_time       | acode | aplaneno
---------+------+---------------------+---------------------+-------+----------
       1 |  350 | 2021-06-27 13:00:00 |                     | IXE   | IG751
       1 | 1000 | 2021-06-27 03:00:00 | 2021-06-27 04:00:00 | BOM   | IG851
       2 | 1100 | 2021-06-27 06:00:00 |                     | KIA   | IG851
       1 | 1300 | 2021-07-14 10:00:00 |                     | VTZ   | SJ100
       1 |  500 | 2021-08-02 18:30:00 | 2021-08-02 19:30:00 | KIA   | AI785
       2 | 1000 | 2021-08-02 21:00:00 |                     | BOM   | AI785
       1 |  400 | 2021-08-07 21:45:00 |                     | IXE   | AA751
       1 | 2100 | 2021-09-17 21:00:00 |                     | IXB   | VT757
       1 | 1400 | 2021-10-10 14:15:00 |                     | DEL   | AA651
       1 | 1110 | 2021-07-07 16:10:25 | 2021-07-07 16:40:00 | KIA   | GF421
       2 | 2273 | 2021-07-07 20:10:25 |                     | JRG   | GF421
       1 | 2556 | 2021-10-12 20:00:00 |                     | IMF   | GF611
       1 | 1252 | 2021-08-02 19:10:00 | 2021-08-02 19:30:00 | AMD   | JA617
       2 | 2220 | 2021-08-02 21:10:00 |                     | KIA   | JA617
       1 | 2605 | 2021-07-14 12:30:00 | 2021-07-14 13:00:00 | BOM   | OG230
       2 | 3115 | 2021-07-14 15:30:00 | 2021-07-14 15:45:00 | IXE   | OG230
       3 | 4001 | 2021-07-14 18:30:00 |                     | KIA   | OG230
       1 | 1015 | 2021-09-17 20:45:30 | 2021-09-17 21:30:00 | KIA   | TJ785
       2 | 2000 | 2021-09-17 17:45:30 |                     | STV   | TJ785
       1 | 1028 | 2021-10-27 06:00:00 |                     | DEL   | KF215
       1 | 1314 | 2021-08-01 22:30:00 |                     | IXB   | OG127
(21 rows)

airres=> truncate arrives_at;
ERROR:  permission denied for table arrives_at
airres=>
```

Developer: What developer looks like without the statement ‘`grant references on all tables in schema public to Developer;`’

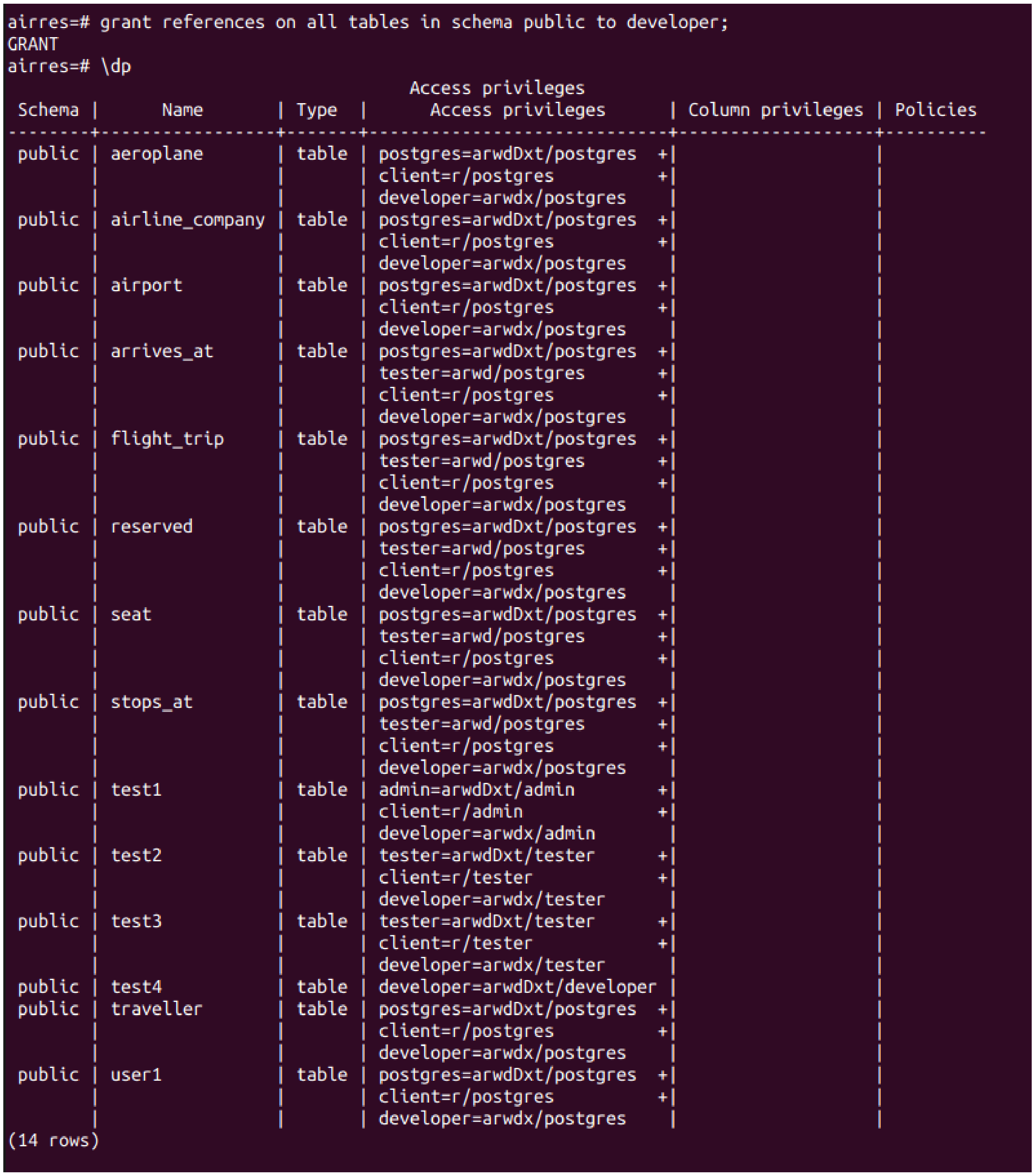


```
airres=> create table test4 (p varchar);
CREATE TABLE
airres=> alter table test4
airres-> add foreign key (p) references airport(acode);
ERROR:  permission denied for table airport
airres=> select * from airport;
                    aname                    | acode |  zip   |          location           |       city        | country |     staete
---------------------------------------------+-------+--------+-----------------------------+-------------------+---------+----------------
 Kempegowda International Airport            | KIA   | 560300 | KIAL Rd Devanahalli         | Bangalore         | India   | Karnataka
 Indira Gandhi International Airport         | DEL   | 110037 | New Delhi Delhi             | New Delhi         | India   | Delhi
 Mangalore International Airport             | IXE   | 574142 | Bajpe Main Rd Kenjar HC     | Mangalore         | India   | Karnataka
 Cochin International Airport                | CIA   | 683111 | Airport Rd Kochi            | Kochi             | India   | Kerala
 Visakhapatnam Airport                       | VTZ   | 530009 | NH 16 Opp Viman Nagar       | Visakhapatnam     | India   | Andhra Pradesh
 Surat International Airport                 | STV   | 395007 | Surat-Dumas Rd Gaviyer      | Surat             | India   | Gujarat
 Bagdogra Airport                            | IXB   | 734421 | Airport Road Bagdogra       | Siliguri          | India   | West Bengal
 Chhatrapati Shivaji International Airport   | BOM   | 400099 | Mumbai Maharashtra          | Mumbai            | India   | Maharashtra
 Sardar Vallabhbhai Patel International Airport | AMD   | 380003 | Hansol, Ahmedabad           | Ahmedabad         | India   | Gujarat
 Kannur International Airport                | CNN   | 670702 | Mattannur, Mattannur Rd.    | Kannur            | India   | Kerala
 Trivandrum International Airport            | TRV   | 695008 | Airport Rd, Chacka          | Thiruvananthapuram | India   | Kerala
 Aurangabad Airport                          | IXU   | 431006 | Jalna Road, Chilkalthana    | Aurangabad        | India   | Maharashtra
 Imphal International Airport                | IMF   | 795140 | Tipaimukh Rd, Hiangtam Lamka | Imphal            | India   | Manipur
 Shillong Airport                            | SHL   | 793103 | Shillong Airport Road       | Umroi             | India   | Meghalaya
 Biju Patnaik International Airport          | BBI   | 751020 | Airport Rd, Aerodrome Area  | Bhubaneswar       | India   | Orissa
 Veer Surendra Sai Airport                   | JRG   | 768204 | SH 10                       | Durlaga           | India   | Orissa
(16 rows)
```



```
airres=> \dp
                         Access privileges
 Schema |      Name       | Type  |     Access privileges      | Co
--------+-----------------+-------+----------------------------+---
 public | aeroplane       | table | postgres=arwdDxt/postgres+|
        |                 |       | client=r/postgres        +|
        |                 |       | developer=arwd/postgres   |
 public | airline_company | table | postgres=arwdDxt/postgres+|
```

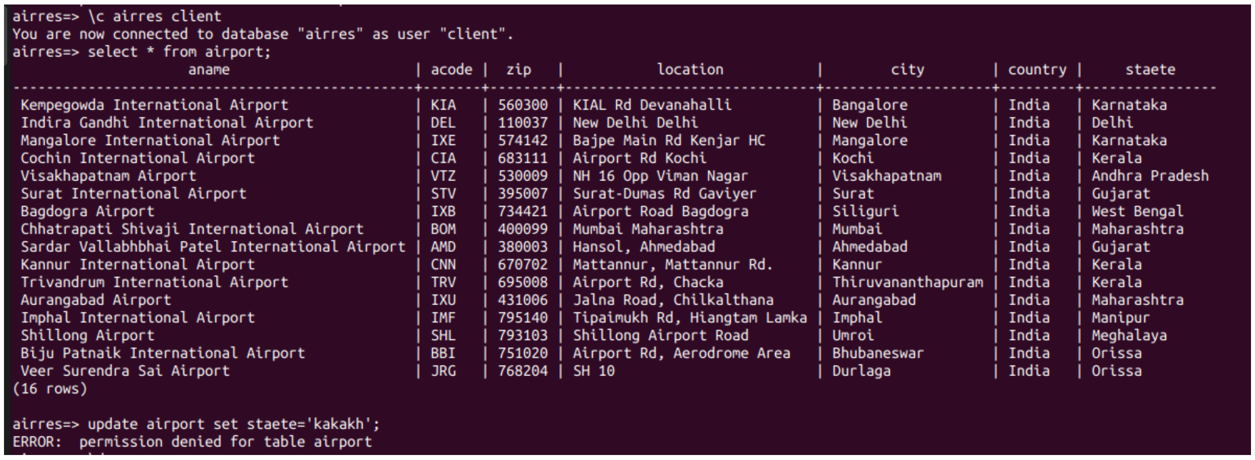
After granting privilege of references:



```
airres=# grant references on all tables in schema public to developer;
GRANT
airres=# \dp
                                     Access privileges
 Schema |      Name       | Type  |       Access privileges       | Column privileges | Policies
--------+-----------------+-------+-------------------------------+-------------------+----------
 public | aeroplane       | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | airline_company | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | airport         | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | arrives_at      | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | tester=arwd/postgres         +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | flight_trip     | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | tester=arwd/postgres         +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | reserved        | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | tester=arwd/postgres         +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | seat            | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | tester=arwd/postgres         +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | stops_at        | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | tester=arwd/postgres         +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | test1           | table | admin=arwdDxt/admin          +|                   |
        |                 |       | client=r/admin               +|                   |
        |                 |       | developer=arwdx/admin         |                   |
 public | test2           | table | tester=arwdDxt/tester        +|                   |
        |                 |       | client=r/tester              +|                   |
        |                 |       | developer=arwdx/tester        |                   |
 public | test3           | table | tester=arwdDxt/tester        +|                   |
        |                 |       | client=r/tester              +|                   |
        |                 |       | developer=arwdx/tester        |                   |
 public | test4           | table | developer=arwdDxt/developer   |                   |
 public | traveller       | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
 public | user1           | table | postgres=arwdDxt/postgres    +|                   |
        |                 |       | client=r/postgres            +|                   |
        |                 |       | developer=arwdx/postgres      |                   |
(14 rows)
```

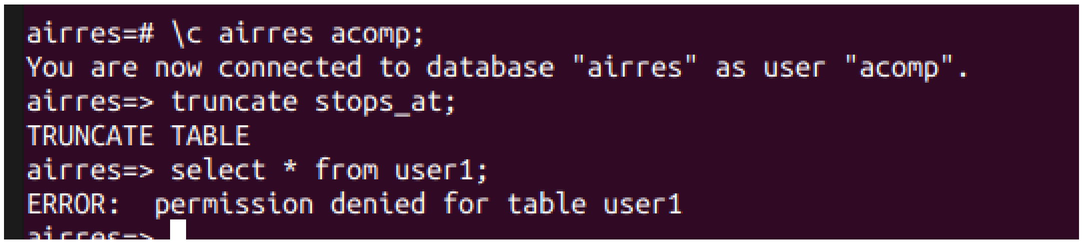
Now, the developer can do almost everything!

<u>Client:</u>



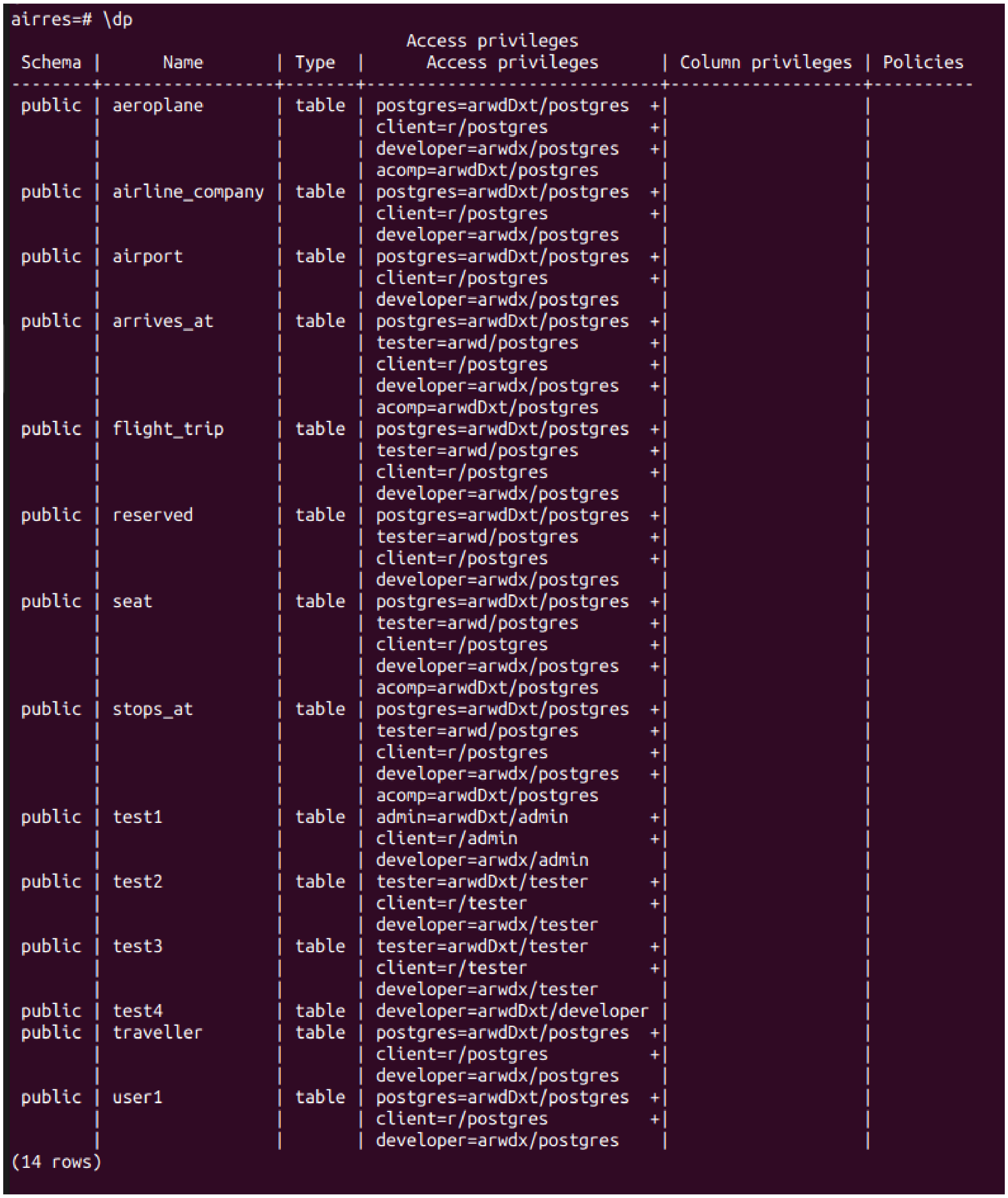
```
airres=> \c airres client
You are now connected to database "airres" as user "client".
airres=> select * from airport;
                     aname                      | acode |  zip   |           location           |        city        | country |     staete
------------------------------------------------+-------+--------+------------------------------+--------------------+---------+----------------
 Kempegowda International Airport               | KIA   | 560300 | KIAL Rd Devanahalli          | Bangalore          | India   | Karnataka
 Indira Gandhi International Airport            | DEL   | 110037 | New Delhi Delhi              | New Delhi          | India   | Delhi
 Mangalore International Airport                | IXE   | 574142 | Bajpe Main Rd Kenjar HC      | Mangalore          | India   | Karnataka
 Cochin International Airport                   | CIA   | 683111 | Airport Rd Kochi             | Kochi              | India   | Kerala
 Visakhapatnam Airport                          | VTZ   | 530009 | NH 16 Opp Viman Nagar        | Visakhapatnam      | India   | Andhra Pradesh
 Surat International Airport                    | STV   | 395007 | Surat-Dumas Rd Gaviyer       | Surat              | India   | Gujarat
 Bagdogra Airport                               | IXB   | 734421 | Airport Road Bagdogra        | Siliguri           | India   | West Bengal
 Chhatrapati Shivaji International Airport      | BOM   | 400099 | Mumbai Maharashtra           | Mumbai             | India   | Maharashtra
 Sardar Vallabhbhai Patel International Airport | AMD   | 380003 | Hansol, Ahmedabad            | Ahmedabad          | India   | Gujarat
 Kannur International Airport                   | CNN   | 670702 | Mattannur, Mattannur Rd.     | Kannur             | India   | Kerala
 Trivandrum International Airport               | TRV   | 695008 | Airport Rd, Chacka           | Thiruvananthapuram | India   | Kerala
 Aurangabad Airport                             | IXU   | 431006 | Jalna Road, Chilkalthana     | Aurangabad         | India   | Maharashtra
 Imphal International Airport                   | IMF   | 795140 | Tipaimukh Rd, Hiangtam Lamka | Imphal             | India   | Manipur
 Shillong Airport                               | SHL   | 793103 | Shillong Airport Road        | Umroi              | India   | Meghalaya
 Biju Patnaik International Airport             | BBI   | 751020 | Airport Rd, Aerodrome Area   | Bhubaneswar        | India   | Orissa
 Veer Surendra Sai Airport                      | JRG   | 768204 | SH 10                        | Durlaga            | India   | Orissa
(16 rows)

airres=> update airport set staete='kakakh';
ERROR:  permission denied for table airport
```

AComp:



```
airres=# \c airres acomp;
You are now connected to database "airres" as user "acomp".
airres=> truncate stops_at;
TRUNCATE TABLE
airres=> select * from user1;
ERROR:  permission denied for table user1
```

Final privilege list:



```
airres=# \dp
                                    Access privileges
 Schema |      Name       | Type  |     Access privileges       | Column privileges | Policies
--------+-----------------+-------+-----------------------------+-------------------+----------
 public | aeroplane       | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres   +|                   |
        |                 |       | acomp=arwdDxt/postgres      |                   |
 public | airline_company | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres    |                   |
 public | airport         | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres    |                   |
 public | arrives_at      | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | tester=arwd/postgres       +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres   +|                   |
        |                 |       | acomp=arwdDxt/postgres      |                   |
 public | flight_trip     | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | tester=arwd/postgres       +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres    |                   |
 public | reserved        | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | tester=arwd/postgres       +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres    |                   |
 public | seat            | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | tester=arwd/postgres       +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres   +|                   |
        |                 |       | acomp=arwdDxt/postgres      |                   |
 public | stops_at        | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | tester=arwd/postgres       +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres   +|                   |
        |                 |       | acomp=arwdDxt/postgres      |                   |
 public | test1           | table | admin=arwdDxt/admin        +|                   |
        |                 |       | client=r/admin             +|                   |
        |                 |       | developer=arwdx/admin       |                   |
 public | test2           | table | tester=arwdDxt/tester      +|                   |
        |                 |       | client=r/tester            +|                   |
        |                 |       | developer=arwdx/tester      |                   |
 public | test3           | table | tester=arwdDxt/tester      +|                   |
        |                 |       | client=r/tester            +|                   |
        |                 |       | developer=arwdx/tester      |                   |
 public | test4           | table | developer=arwdDxt/developer |                   |
 public | traveller       | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres    |                   |
 public | user1           | table | postgres=arwdDxt/postgres  +|                   |
        |                 |       | client=r/postgres          +|                   |
        |                 |       | developer=arwdx/postgres    |                   |
(14 rows)
```

TRANSACTIONS AND CONCURRENCY

A database transaction symbolizes a unit of work performed within a database management system against a database, and treated in a coherent and reliable way independent of other transactions.

A transaction generally represents any change in a database. Database concurrency is the ability of a database to allow multiple users to affect multiple transactions. The ability to offer concurrency is unique to databases.

Concurrency control methods in postgres:

Concurrency Control is a mechanism that maintains atomicity and isolation, which are two properties of the ACID, when several transactions run concurrently in the database.

There are three broad concurrency control techniques, i.e. *Multi-version Concurrency Control* (MVCC), *Strict Two-Phase Locking* (S2PL), and *Optimistic Concurrency Control* (OCC), and each technique has many variations.

In MVCC, each write operation creates a new version of a data item while retaining the old version. When a transaction reads a data item, the system selects one of the versions to ensure isolation of the individual transaction.

The main advantage of MVCC is that '*readers don't block writers, and writers don't block readers*', in contrast, for example, an S2PL-based system must block readers when a writer writes an item because the writer acquires an exclusive lock for the item.

PostgreSQL and some RDBMSs use a variation of MVCC called **Snapshot Isolation (SI)**.

Phenomenon prohibited in Postgres:

The phenomena which are prohibited at various levels are:

dirty read

A transaction reads data written by a concurrent uncommitted transaction.

nonrepeatable read

A transaction re-reads data it has previously read and finds that data has been modified by another transaction (that committed since the initial read).

phantom read

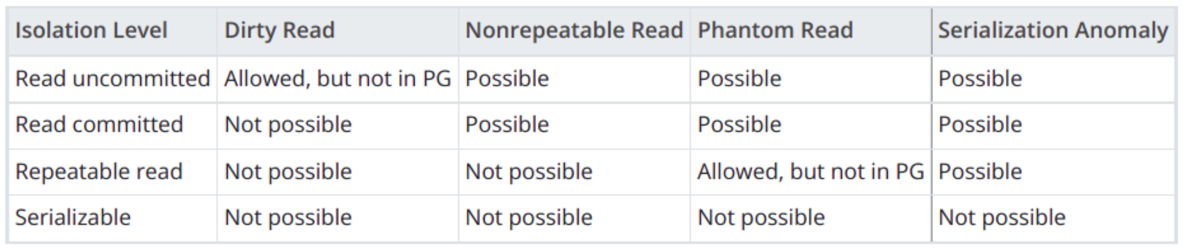
A transaction re-executes a query returning a set of rows that satisfy a search condition and finds that the set of rows satisfying the condition has changed due to another recently-committed transaction.

serialization anomaly

The result of successfully committing a group of transactions is inconsistent with all possible orderings of running those transactions one at a time.

These are taken care of by using the following levels of isolation:

Table 13.1. Transaction Isolation Levels

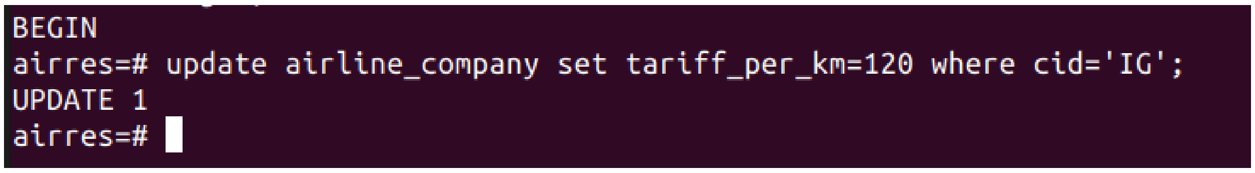


Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

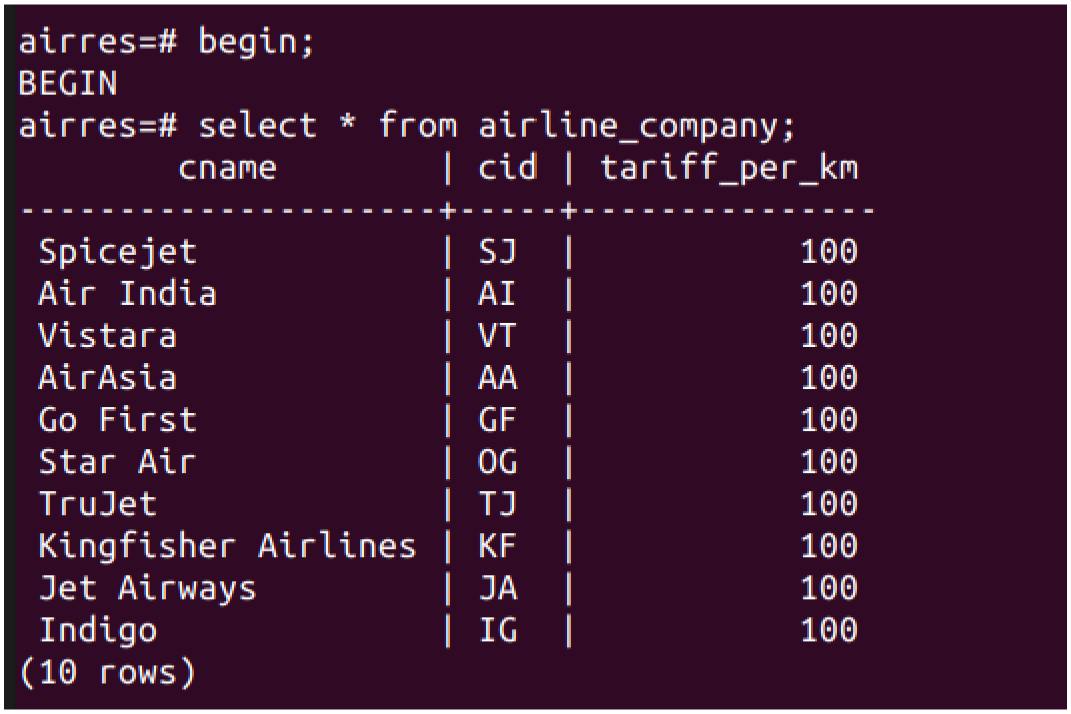
There are four isolation levels defined by the standard: *read uncommitted*, *read committed*, *repeatable read*, and *serializable*. PostgreSQL doesn't implement *read uncommitted*, which allows *dirty reads*, and instead defaults to *read committed*.

Demo for Read Committed Isolation Level

On psql session 1:

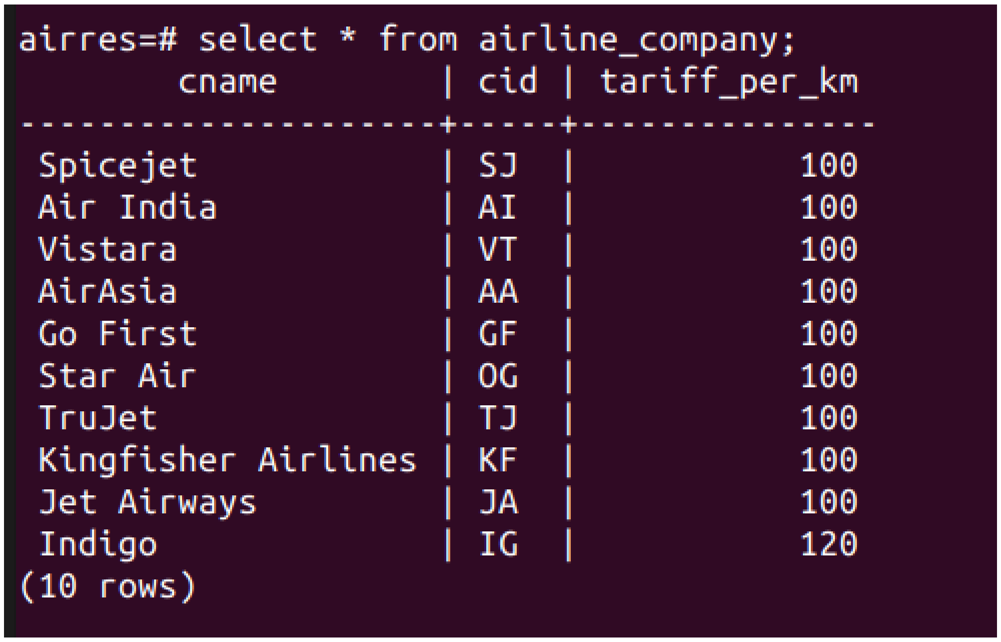

```
BEGIN
airres=# update airline_company set tariff_per_km=120 where cid='IG';
UPDATE 1
airres=#
```

Psql session 2:


```
airres=# begin;
BEGIN
airres=# select * from airline_company;
        cname         | cid | tariff_per_km
----------------------+-----+---------------
 Spicejet             | SJ  |           100
 Air India            | AI  |           100
 Vistara              | VT  |           100
 AirAsia              | AA  |           100
 Go First             | GF  |           100
 Star Air             | OG  |           100
 TruJet               | TJ  |           100
 Kingfisher Airlines  | KF  |           100
 Jet Airways          | JA  |           100
 Indigo               | IG  |           100
(10 rows)
```

You can see that this new transaction query uses on earlier snapshot rather than reading uncommitted data

The same transaction, the same query after transaction 1 is committed:

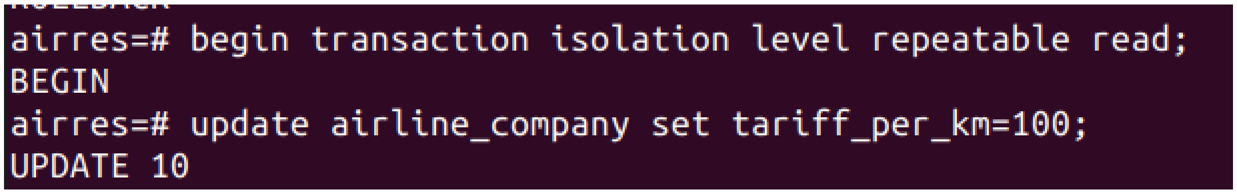


```
airres=# select * from airline_company;
        cname         | cid | tariff_per_km
----------------------+-----+---------------
 Spicejet             | SJ  |           100
 Air India            | AI  |           100
 Vistara              | VT  |           100
 AirAsia              | AA  |           100
 Go First             | GF  |           100
 Star Air             | OG  |           100
 TruJet               | TJ  |           100
 Kingfisher Airlines  | KF  |           100
 Jet Airways          | JA  |           100
 Indigo               | IG  |           120
(10 rows)
```

<u>**Example for Repeatable Read Isolation Level:**</u>

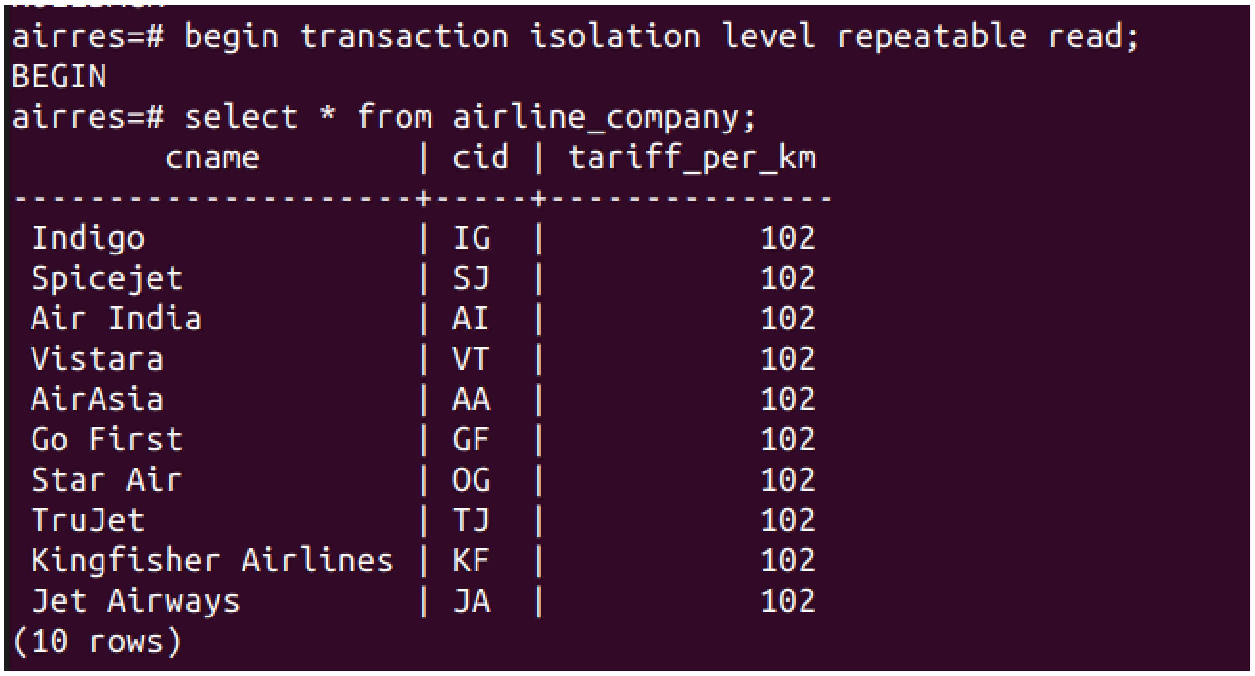
This level of isolation is implemented using locks:

On session 1:



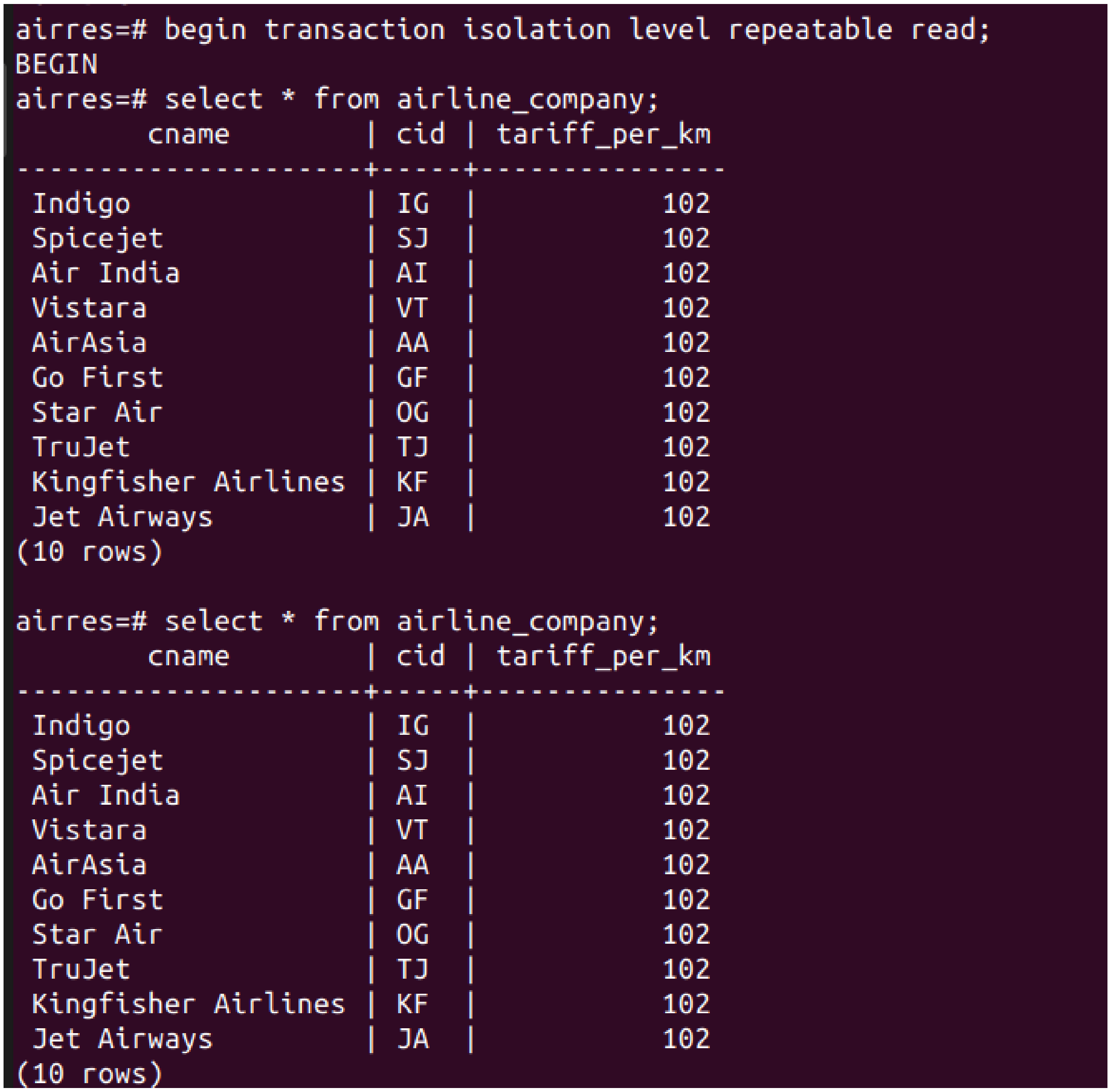
```
airres=# begin transaction isolation level repeatable read;
BEGIN
airres=# update airline_company set tariff_per_km=100;
UPDATE 10
```

Session 2:



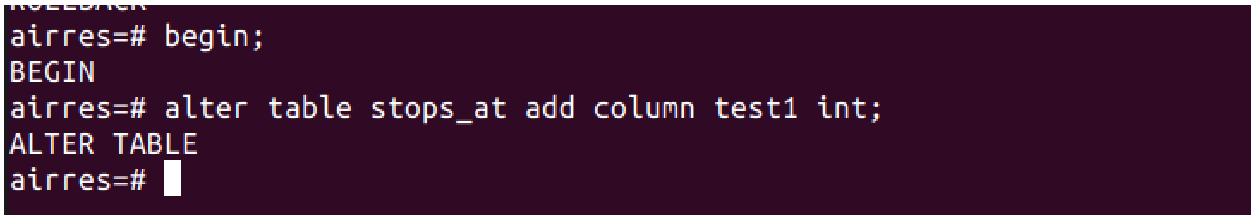
```
airres=# begin transaction isolation level repeatable read;
BEGIN
airres=# select * from airline_company;
        cname        | cid | tariff_per_km
---------------------+-----+---------------
 Indigo              | IG  |           102
 Spicejet            | SJ  |           102
 Air India           | AI  |           102
 Vistara             | VT  |           102
 AirAsia             | AA  |           102
 Go First            | GF  |           102
 Star Air            | OG  |           102
 TruJet              | TJ  |           102
 Kingfisher Airlines | KF  |           102
 Jet Airways         | JA  |           102
(10 rows)
```

After session 1 is committed, session 2:



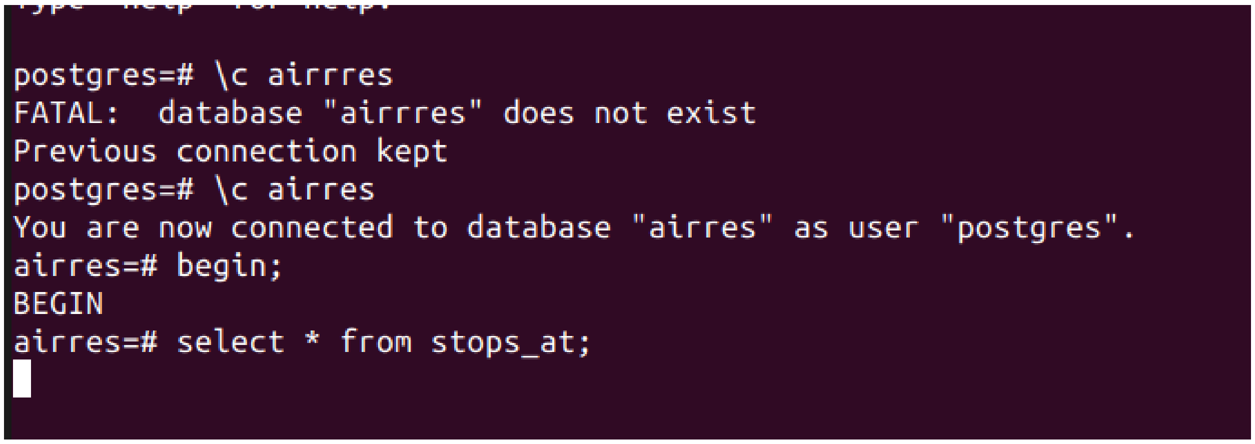
```
airres=# begin transaction isolation level repeatable read;
BEGIN
airres=# select * from airline_company;
        cname         | cid | tariff_per_km
----------------------+-----+---------------
 Indigo               | IG  |           102
 Spicejet             | SJ  |           102
 Air India            | AI  |           102
 Vistara              | VT  |           102
 AirAsia              | AA  |           102
 Go First             | GF  |           102
 Star Air             | OG  |           102
 TruJet               | TJ  |           102
 Kingfisher Airlines  | KF  |           102
 Jet Airways          | JA  |           102
(10 rows)

airres=# select * from airline_company;
        cname         | cid | tariff_per_km
----------------------+-----+---------------
 Indigo               | IG  |           102
 Spicejet             | SJ  |           102
 Air India            | AI  |           102
 Vistara              | VT  |           102
 AirAsia              | AA  |           102
 Go First             | GF  |           102
 Star Air             | OG  |           102
 TruJet               | TJ  |           102
 Kingfisher Airlines  | KF  |           102
 Jet Airways          | JA  |           102
(10 rows)
```

Another Demo For locks:



```
airres=# begin;
BEGIN
airres=# alter table stops_at add column test1 int;
ALTER TABLE
airres=#
```

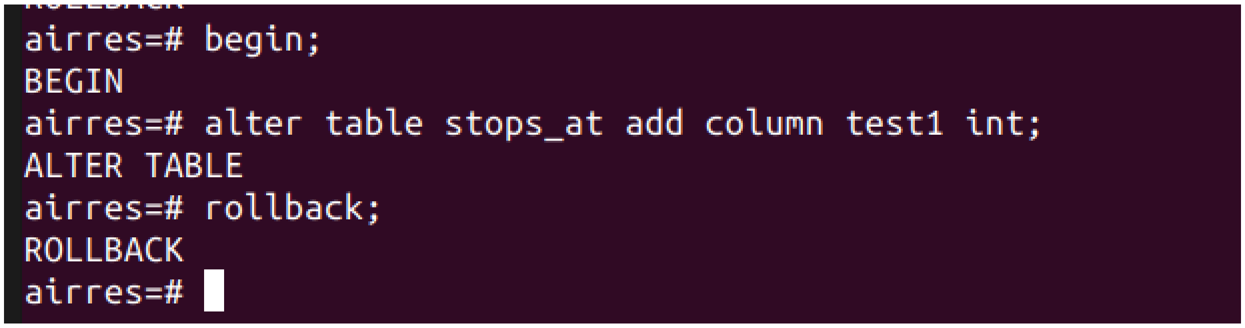
Simultaneously, on another prompt, we run the select query on the same table



```
postgres=# \c airrres
FATAL:  database "airrres" does not exist
Previous connection kept
postgres=# \c airres
You are now connected to database "airres" as user "postgres".
airres=# begin;
BEGIN
airres=# select * from stops_at;
```

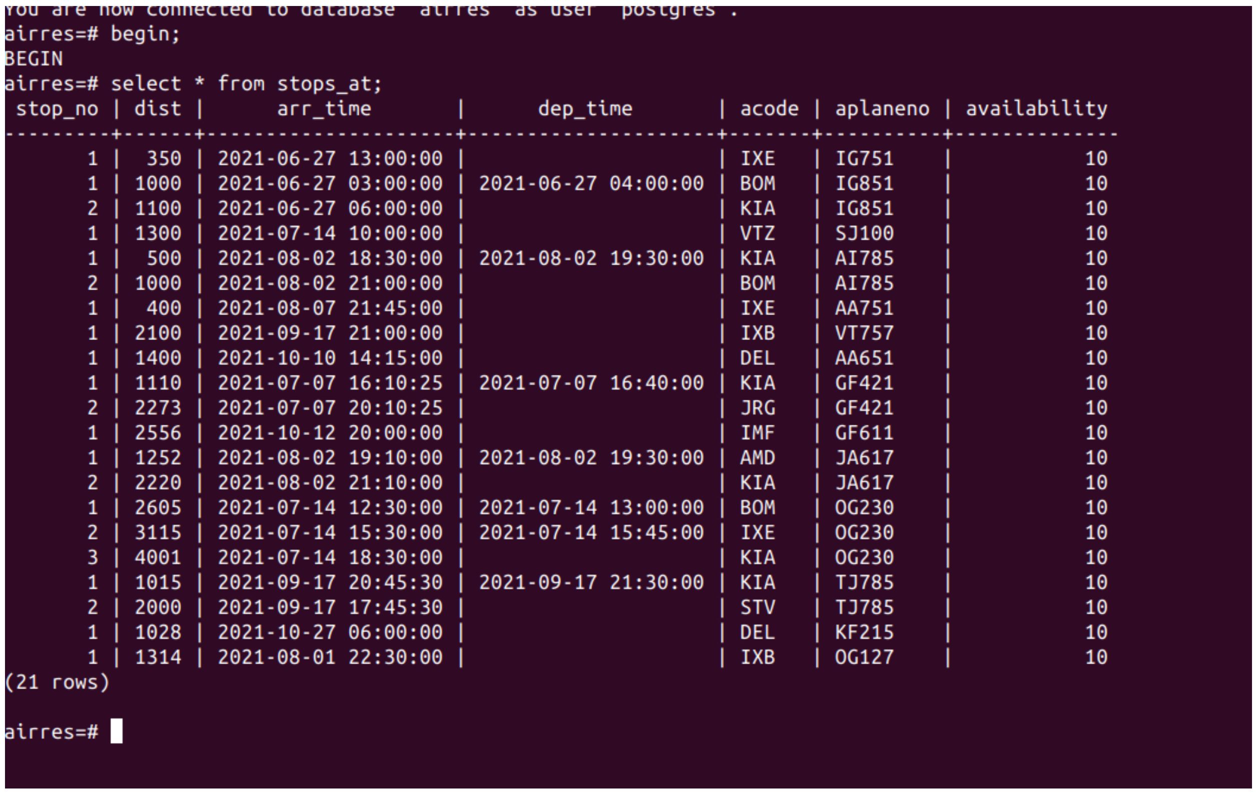
This query will not execute until the first transaction is committed in some form.

The minute rollback is issued:



```
airres=# begin;
BEGIN
airres=# alter table stops_at add column test1 int;
ALTER TABLE
airres=# rollback;
ROLLBACK
airres=#
```

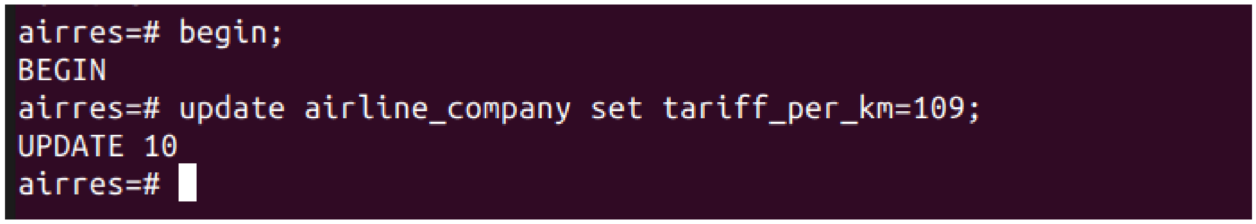
The second transaction gives a result:



```
You are now connected to database "airres" as user "postgres".
airres=# begin;
BEGIN
airres=# select * from stops_at;
 stop_no | dist |      arr_time       |      dep_time       | acode | aplaneno | availability
---------+------+---------------------+---------------------+-------+----------+-------------
       1 |  350 | 2021-06-27 13:00:00 |                     | IXE   | IG751    |           10
       1 | 1000 | 2021-06-27 03:00:00 | 2021-06-27 04:00:00 | BOM   | IG851    |           10
       2 | 1100 | 2021-06-27 06:00:00 |                     | KIA   | IG851    |           10
       1 | 1300 | 2021-07-14 10:00:00 |                     | VTZ   | SJ100    |           10
       1 |  500 | 2021-08-02 18:30:00 | 2021-08-02 19:30:00 | KIA   | AI785    |           10
       2 | 1000 | 2021-08-02 21:00:00 |                     | BOM   | AI785    |           10
       1 |  400 | 2021-08-07 21:45:00 |                     | IXE   | AA751    |           10
       1 | 2100 | 2021-09-17 21:00:00 |                     | IXB   | VT757    |           10
       1 | 1400 | 2021-10-10 14:15:00 |                     | DEL   | AA651    |           10
       1 | 1110 | 2021-07-07 16:10:25 | 2021-07-07 16:40:00 | KIA   | GF421    |           10
       2 | 2273 | 2021-07-07 20:10:25 |                     | JRG   | GF421    |           10
       1 | 2556 | 2021-10-12 20:00:00 |                     | IMF   | GF611    |           10
       1 | 1252 | 2021-08-02 19:10:00 | 2021-08-02 19:30:00 | AMD   | JA617    |           10
       2 | 2220 | 2021-08-02 21:10:00 |                     | KIA   | JA617    |           10
       1 | 2605 | 2021-07-14 12:30:00 | 2021-07-14 13:00:00 | BOM   | OG230    |           10
       2 | 3115 | 2021-07-14 15:30:00 | 2021-07-14 15:45:00 | IXE   | OG230    |           10
       3 | 4001 | 2021-07-14 18:30:00 |                     | KIA   | OG230    |           10
       1 | 1015 | 2021-09-17 20:45:30 | 2021-09-17 21:30:00 | KIA   | TJ785    |           10
       2 | 2000 | 2021-09-17 17:45:30 |                     | STV   | TJ785    |           10
       1 | 1028 | 2021-10-27 06:00:00 |                     | DEL   | KF215    |           10
       1 | 1314 | 2021-08-01 22:30:00 |                     | IXB   | OG127    |           10
(21 rows)

airres=# 
```

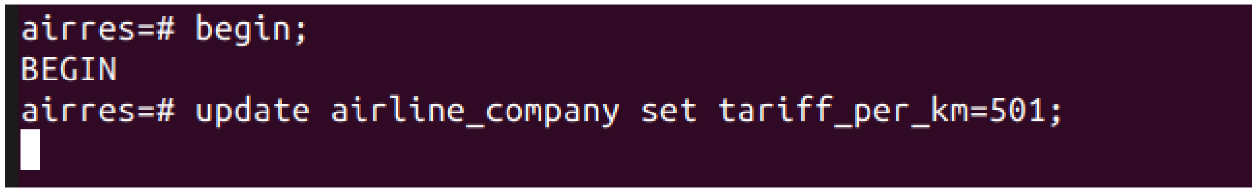
Demo for blocking lost update:

Session 1:



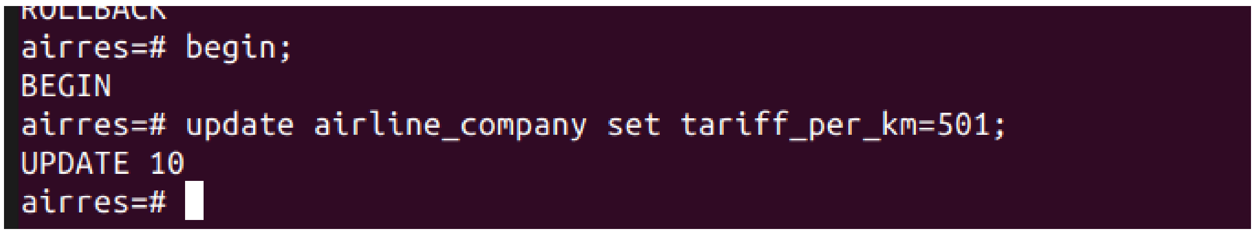
```
airres=# begin;
BEGIN
airres=# update airline_company set tariff_per_km=109;
UPDATE 10
airres=# 
```

Session 2:



```
airres=# begin;
BEGIN
airres=# update airline_company set tariff_per_km=501;
```

Once session 1 is complete, session 2:



```
ROLLBACK
airres=# begin;
BEGIN
airres=# update airline_company set tariff_per_km=501;
UPDATE 10
airres=#
```

Contributions

Rithika Pai

-

Rithika Shankar

- Transactions
- Report

1 hour

Ramya Prabhu

- Simple Queries
- Complex Queries
- Execution Plan and Performance
- Multiple users
- Transactions
- [Report for the same]

4 hours