

Laboratorio 08

1. Ejercicio 1 – selectividad de los atributos.

USUARIO	DETALLE
ID de usuario	ID del usuario
Número de serie de la báscula	Fecha y hora de cada pesaje
Fecha de nacimiento	Valor en libras del peso registrado.
Fecha del inicio de uso de servicio	
Altura	
Peso inicial al momento de uso del servicio.	
País de nacimiento	

2. Ejercicio 2 – Declaración de índices.

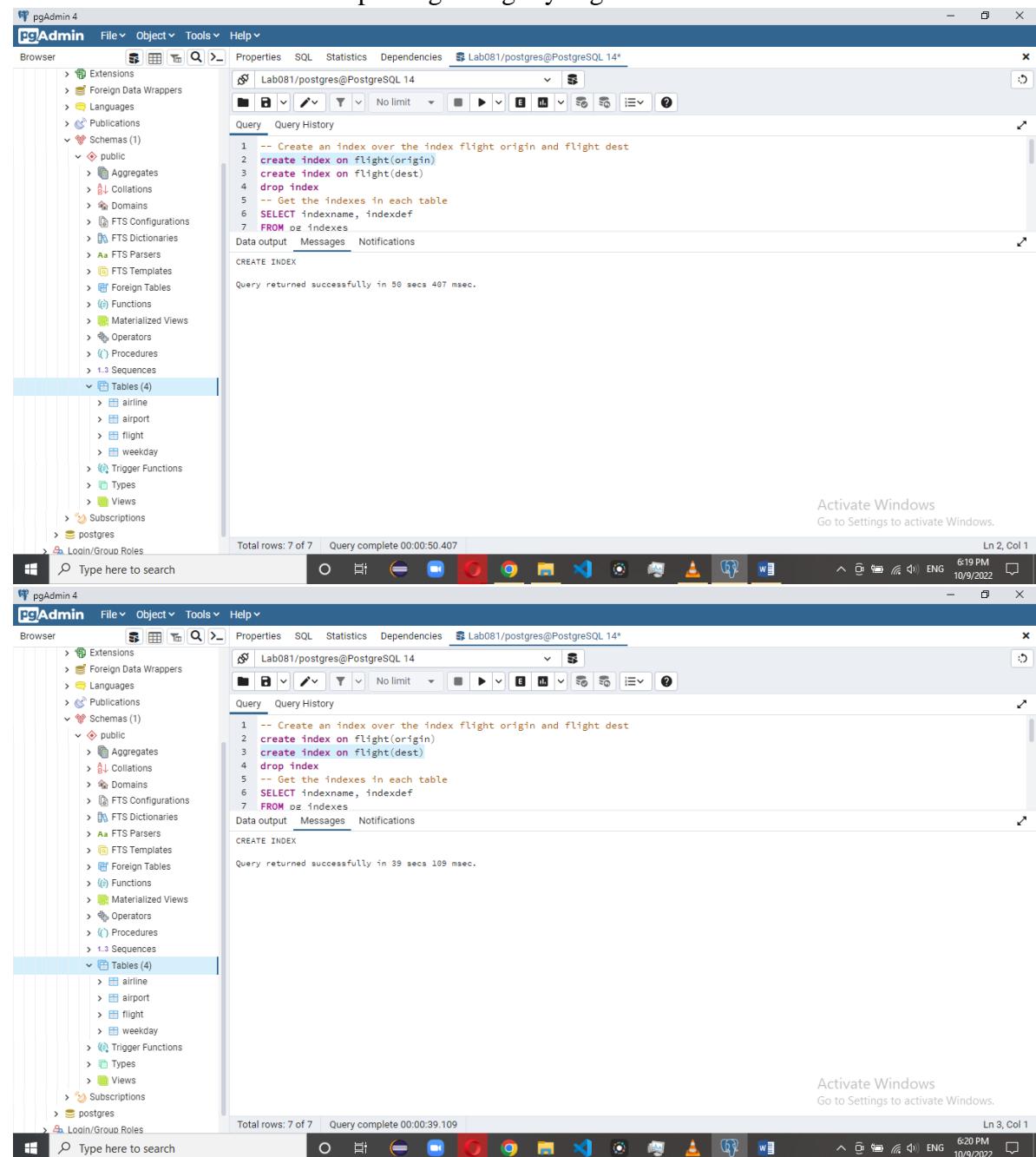
- a. Investigue cómo enumerar los índices existentes en la base de datos y muéstrelas a continuación.

```

-- Get the indexes in each table
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'flight' OR tablename = 'airline' OR tablename = 'airport' OR tablename = 'weekday';

```

indexname	indexdef
idx_123793_sqlite_autoindex_weekdays_1	CREATE UNIQUE INDEX idx_123793_sqlite_autoindex_weekdays_1 ON public.weekday USING btree (dayofweek)
idx_123781_date	CREATE INDEX idx_123781_date ON public.flight USING btree (year, month, dayofmonth)
idx_123781_dest	CREATE INDEX idx_123781_dest ON public.flight USING btree (dest)
idx_123781_index_1	CREATE INDEX idx_123781_index_1 ON public.flight USING btree (origin)
idx_123781_origin	CREATE INDEX idx_123781_origin ON public.flight USING btree (origin)
idx_123781_year	CREATE INDEX idx_123781_year ON public.flight USING btree (year)
idx_123787_index_2	CREATE INDEX idx_123787_index_2 ON public.airport USING btree (lata)

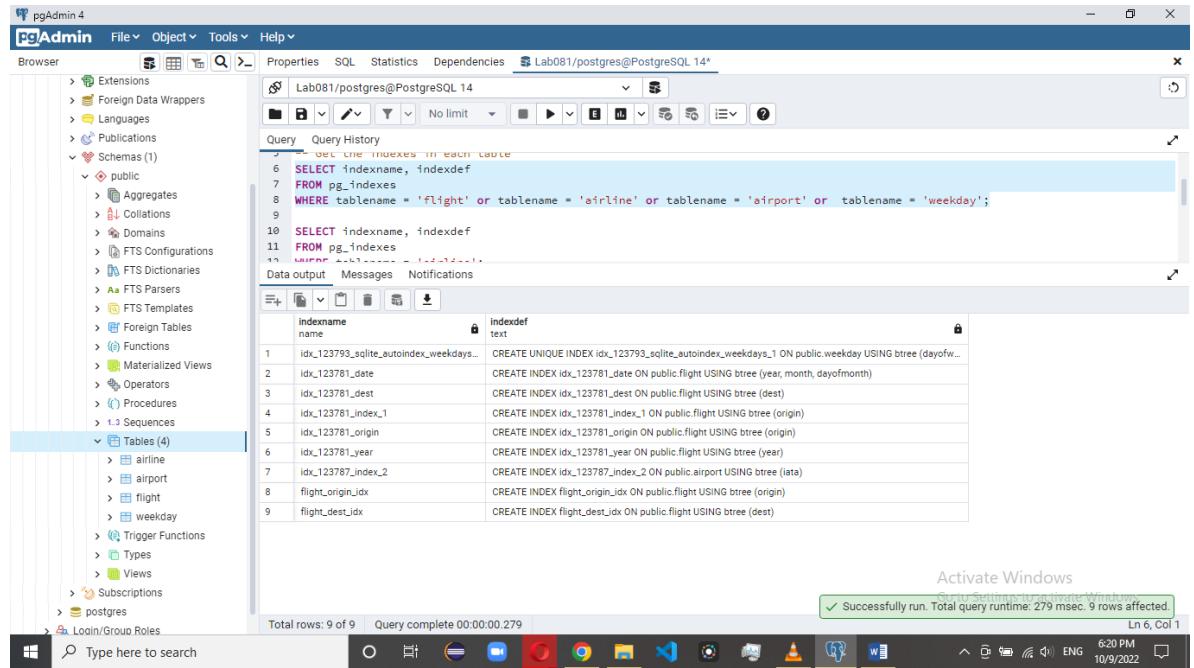
b. Defina un índice sobre los campos flight.origin y flight.dest

```
-- Create an index over the index flight origin and flight dest
create index on flight(origin)
create index on flight(dest)
drop index
-- Get the indexes in each table
SELECT indexname, indexdef
FROM oz_indexes
Data output Messages Notifications
CREATE INDEX
Query returned successfully in 50 secs 407 msec.
```

Total rows: 7 of 7 Query complete 00:00:50.407

```
-- Create an index over the index flight origin and flight dest
create index on flight(origin)
create index on flight(dest)
drop index
-- Get the indexes in each table
SELECT indexname, indexdef
FROM oz_indexes
Data output Messages Notifications
CREATE INDEX
Query returned successfully in 39 secs 109 msec.
```

Total rows: 7 of 7 Query complete 00:00:39.109



```

pgAdmin 4
File Object Tools Help
Lab081/postgres@PostgreSQL 14*
Browser Properties SQL Statistics Dependencies
Query Query History
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'Flight' OR tablename = 'airline' OR tablename = 'airport' OR tablename = 'weekday';
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'Flight' OR tablename = 'airline' OR tablename = 'airport' OR tablename = 'weekday';

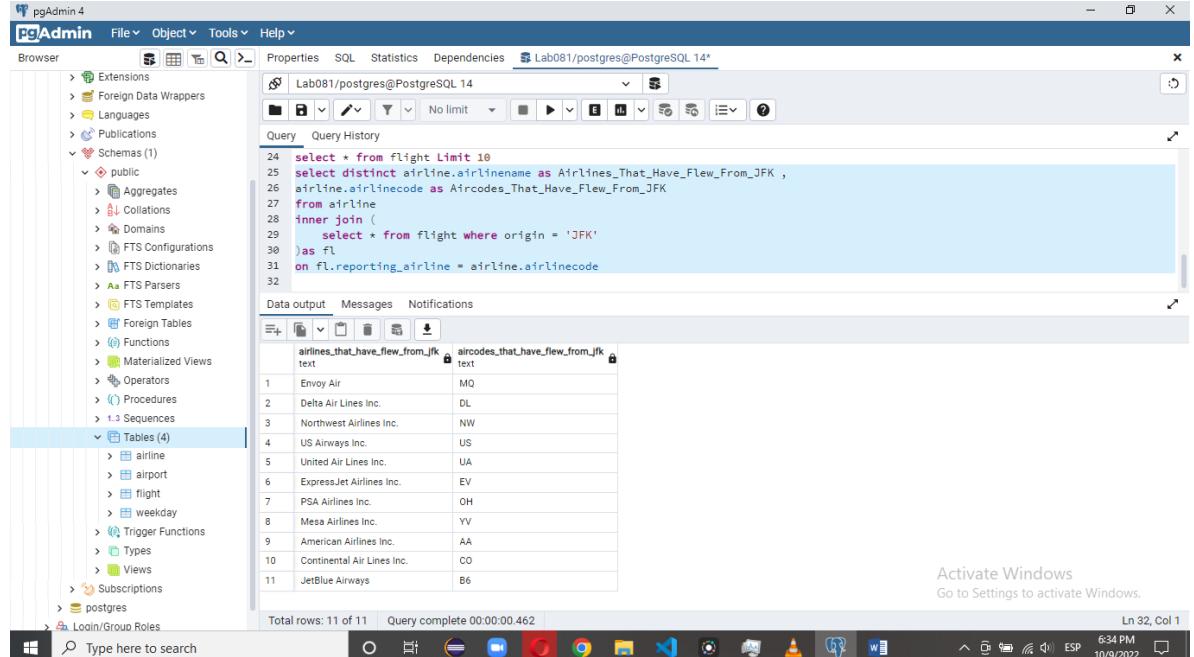
Data output Messages Notifications
Indexname name indexdef
idx_123793_sqlite_autoindex_weekdays_1 ON public.weekday USING btree (dayofweek)
idx_123781_date ON public.flight USING btree (year, month, dayofmonth)
idx_123781_dest ON public.flight USING btree (dest)
idx_123781_index_1 ON public.flight USING btree (origin)
idx_123781_origin ON public.flight USING btree (origin)
idx_123781_year ON public.flight USING btree (year)
idx_123787_index_2 ON public.airport USING btree (lata)
flight_origin_idx ON public.flight USING btree (origin)
flight_dest_idx ON public.flight USING btree (dest)

Total rows: 9 of 9 Query complete 00:00:00.279
Activate Windows
Successfully run. Total query runtime: 279 msec. 9 rows affected.
Ln 6, Col 1

```

Como pueden ver se crearon esos índices.

- c. Query con los códigos de las aerolíneas que realizan vuelos que salen del aeropuerto JFK (sin duplicados), e indique cuanto tiempo toma la ejecución de su query.



```

pgAdmin 4
File Object Tools Help
Lab081/postgres@PostgreSQL 14*
Browser Properties SQL Statistics Dependencies
Query Query History
select * from flight Limit 10
select distinct airline.airlinename as Airlines_That_Have_Flew_From_JFK ,
airline.airlinecode as Aircodes_That_Have_Flew_From_JFK
from airline
inner join (
select * from flight where origin = 'JFK'
) as fl
on fl.reporting_airline = airline.airlinecode

Data output Messages Notifications
airlines_that_have_flew_from_jfk aircodes_that_have_flew_from_jfk
text text
1 Envoy Air MQ
2 Delta Air Lines Inc. DL
3 Northwest Airlines Inc. NW
4 US Airways Inc. US
5 United Air Lines Inc. UA
6 ExpressJet Airlines Inc. EV
7 PSA Airlines Inc. OH
8 Mesa Airlines Inc. YV
9 American Airlines Inc. AA
10 Continental Air Lines Inc. CO
11 JetBlue Airways B6

Total rows: 11 of 11 Query complete 00:00:00.462
Activate Windows
Go to Settings to activate Windows.
Ln 32, Col 1

```

Nombres de las aerolíneas y códigos de las aerolíneas, se tardó 462 ms.

The screenshot shows the pgAdmin 4 interface with a query window containing the following SQL code:

```

23 select unique airlinename from airline
24 select * from flight Limit 10
25 select distinct airline.airlinecode as Aircodes_That_Have_Flew_From_JFK
26 from airline
27 inner join (
28   select * from flight where origin = 'JFK'
29 )as fl
30 on fl.reporting_airline = airline.airlinecode
31
  
```

The results table displays 11 rows of airline codes:

	Aircodes_That_Have_Flew_From_JFK
1	YY
2	AA
3	US
4	NW
5	CO
6	B6
7	DL
8	OH
9	EV
10	MQ
11	UA

Total rows: 11 of 11 | Query complete 00:00:00.291

Código de las aerolíneas, se tardó 291 ms.

The screenshot shows the pgAdmin 4 interface with a query window containing the same SQL code as the previous screenshot:

```

23 select unique airlinename from airline
24 select * from flight Limit 10
25 select distinct airline.airlinename as Airlines_That_Have_Flew_From_JFK
26 from airline
27 inner join (
28   select * from flight where origin = 'JFK'
29 )as fl
30 on fl.reporting_airline = airline.airlinecode
31
  
```

The results table displays 11 rows of airline names:

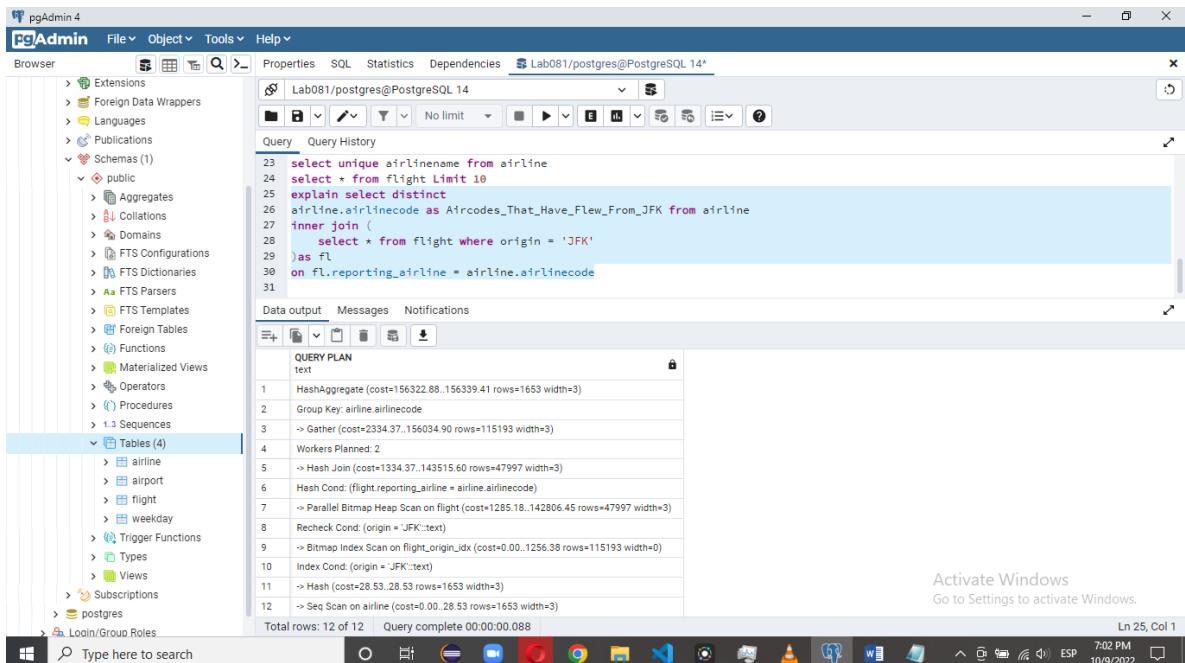
	Airlines_That_Have_Flew_From_JFK
1	American Airlines Inc.
2	Continental Air Lines Inc.
3	US Airways Inc.
4	ExpressJet Airlines Inc.
5	Northwest Airlines Inc.
6	PSA Airlines Inc.
7	JetBlue Airways
8	United Air Lines Inc.
9	Mesa Airlines Inc.
10	Delta Air Lines Inc.
11	Envoy Air

Total rows: 11 of 11 | Query complete 00:00:00.299

Activate Windows
Go to Settings to activate Windows.
Ln 30, Col 46

Nombre de las aerolíneas, se tardó 299 ms.

- Utilizando explain, muestre qué índice se utiliza para optimizar la ejecución de esa consulta.



The screenshot shows the pgAdmin 4 interface with a query editor and a query plan window. The query is:

```

23 select unique airlinename from airline
24 select * from flight Limit 10
25 explain select distinct
26 airline.airlinecode as Aircodes_That_Have_Flew_From_JFK from airline
27 inner join (
28   select * from flight where origin = 'JFK'
29 ) as fL
30 on fL.reporting_airline = airline.airlinecode
31

```

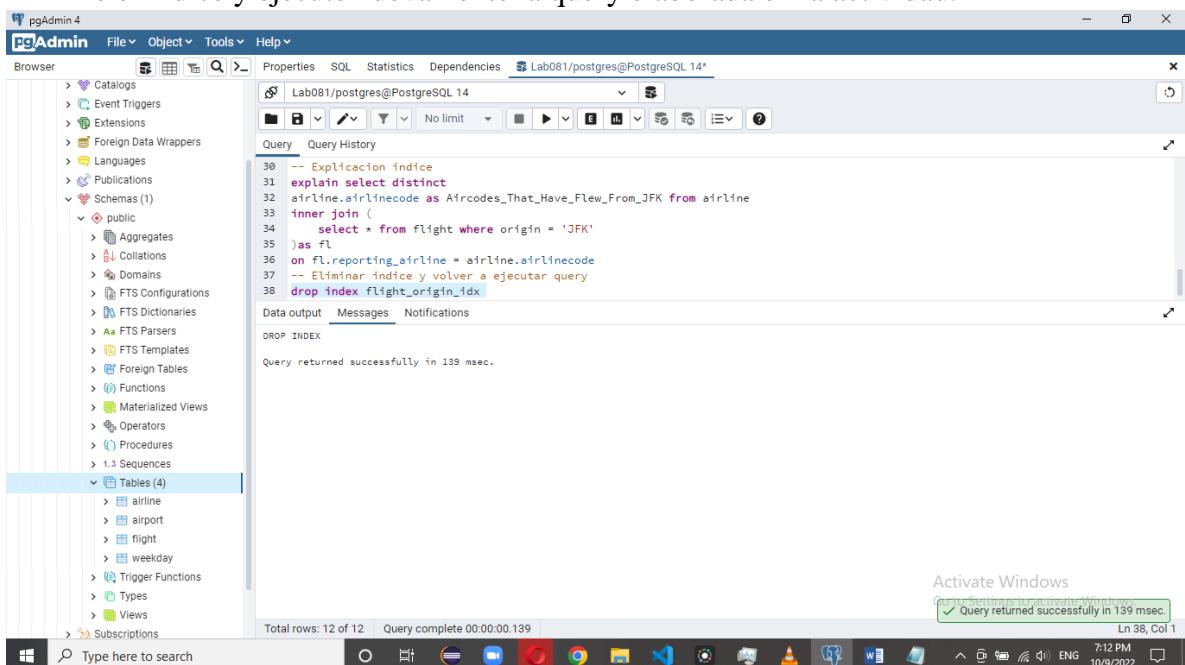
The query plan window displays the execution plan for this query, showing the following steps:

- HashAggregate (cost=156322.88..156339.41 rows=1653 width=3)
- Group Key: airline.airlinecode
- >- Gather (cost=2334.37..156034.90 rows=115193 width=3)
- Workers Planned: 2
- >- Hash Join (cost=1334.37..143515.60 rows=47997 width=3)
- >- Hash Cond: (flight.reporting_airline = airline.airlinecode)
- >- Parallel Bitmap Heap Scan on flight (cost=1285.18..142806.45 rows=47997 width=3)
- Recheck Cond: (origin = 'JFK':text)
- >- Bitmap Index Scan on flight_origin_idx (cost=0.00..1256.38 rows=115193 width=0)
- Index Cond: (origin = 'JFK':text)
- >- Hash (cost=28.53..28.53 rows=1653 width=3)
- >- Seq Scan on airline (cost=0.00..28.53 rows=1653 width=3)
- >- Seq Scan on airline (cost=0.00..28.53 rows=1653 width=3)

Total rows: 12 of 12 | Query complete 00:00:00.088

Se utiliza el índice flight_origin_idx para optimizar la búsqueda.

- e. Elimine el índice y ejecute nuevamente la query elaborada en la actividad.



The screenshot shows the pgAdmin 4 interface with a query editor. The query is:

```

30 -- Explicación indice
31 explain select distinct
32 airline.airlinecode as Aircodes_That_Have_Flew_From_JFK from airline
33 inner join (
34   select * from flight where origin = 'JFK'
35 ) as fL
36 on fL.reporting_airline = airline.airlinecode
37 -- Eliminar indice y volver a ejecutar query
38 drop index flight_origin_idx

```

The query plan window shows the execution plan for the query, which includes the drop index command. The status bar at the bottom right indicates "Query returned successfully in 139 msec."

Total rows: 12 of 12 | Query complete 00:00:00.139

Eliminación del índice

```

36 on fl.reporting_airline = airline.airlinecode
37 -- Eliminar indice y volver a ejecutar query
38 drop index flight_origin_idx
39 select distinct
40 airline.airlinecode as Aircodes_That_Have_Flew_From_JFK from airline
41 inner join (
42 select * from flight where origin = 'JFK'
43 )as fl
44 on fl.reporting_airline = airline.airlinecode
    
```

	aircodes_that_have_flew_from_jfk
1	YV
2	AA
3	US
4	NW
5	CO
6	B6
7	DL
8	OH
9	EV
10	MQ
11	UA

Total rows: 11 of 11 | Query complete 00:00:00.421

Ejecución de la query nuevamente, se ejecutó en 421 ms, casi lo que se tarda en hacer para dos columnas.

- f. Cree nuevamente el índice eliminado.

```

34 select * from flight where origin = 'JFK'
35 )as fl
36 on fl.reporting_airline = airline.airlinecode
37 -- Eliminar indice y volver a ejecutar query
38 drop index flight_origin_idx
39 select distinct
40 airline.airlinecode as Aircodes_That_Have_Flew_From_JFK from airline
41 inner join (
42 select * from flight where origin = 'JFK'
43 )as fl
44 on fl.reporting_airline = airline.airlinecode
45 -- Volver a crear el indice
46 create index on flight(origin)
47 -- Ejecutar la query y tomar su tiempo de ejecucion
    
```

CREATE INDEX

Query returned successfully in 42 secs 687 msec.

Total rows: 11 of 11 | Query complete 00:00:42.687

3. Ejercicio 3 – Selección de índices.

- a. Ejecute el query e indique el tiempo de su ejecución

```

44  on fl.reporting_airline = airline.airlinecode
45  -- Volver a crear el indice
46  create index on flight(origin)
47  -- Ejecutar la query y tomar su tiempo de ejecucion
48  SELECT a.airlinecode, AVG(f.arrdelay) AS arr_delay_average
49  FROM flight f
50  LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
51  WHERE a.airlinecode IN (
52    SELECT DISTINCT f.reporting_airline
53    FROM flight f
54    WHERE f.dest = 'JFK'
55  )
56  GROUP BY a.airlinecode
57  ORDER BY arr_delay_average DESC;

```

airlinecode	arr_delay_average
AA	12.607194035713981
OH	11.81746768998001
YY	11.775181433600801
UA	11.291322186680181
B6	11.084184390518121
CO	10.979037280291301
EV	10.208002415713781
MQ	9.8906679457764961

Total rows: 11 of 11 Query complete 00:00:07.201

Se tardó 7.201 segundos en ejecutarse

- b. Ejecute un explain sobre este query e identifique el costo relativo del query completo reportado por PostgreSQL

```

"Sort (cost=409409.44..409413.57 rows=1653 width=35)"
"  Sort Key: (avg(f.arrdelay)) DESC"
"  -> HashAggregate (cost=409300.41..409321.08 rows=1653
width=35)"
"    Group Key: a.airlinecode"
"    -> Hash Join (cost=160467.89..374251.77 rows=7009728
width=7)"
"      Hash Cond: (f.reporting_airline = a.airlinecode)"
"      -> Seq Scan on flight f (cost=0.00..186649.28 rows=7009728
width=7)"
"      -> Hash (cost=160467.64..160467.64 rows=20 width=6)"
"        -> Merge Join (cost=160459.08..160467.64 rows=20
width=6)"
"          Merge Cond: (a.airlinecode = f_1.reporting_airline)"
"          -> Sort (cost=116.89..121.02 rows=1653 width=3)"
"            Sort Key: a.airlinecode"
"            -> Seq Scan on airline a (cost=0.00..28.53
rows=1653 width=3)"
"              -> Sort (cost=160342.19..160342.24 rows=20
width=3)"
"                Sort Key: f_1.reporting_airline"
"                -> HashAggregate (cost=160341.36..160341.56
rows=20 width=3)"

```

"	Group Key: f_1.reporting_airline"
"	-> Gather (cost=2456.70..160014.82 rows=130615 width=3)"
"	Workers Planned: 2"
"	-> Parallel Bitmap Heap Scan on flight f_1 (cost=1456.70..145953.32 rows=54423 width=3)"
"	Recheck Cond: (dest = 'JFK'::text)"
"	-> Bitmap Index Scan on flight_dest_idx (cost=0.00..1424.04 rows=130615 width=0)"
"	Index Cond: (dest = 'JFK'::text)"

Los costos relativos fueron:

- 0.00..1424.04 para el bitmap index scan.
- 1456.70..145953.32 para el bitmap heap scan paralelo.
- 2456.70..160014.82 para la recolección.
- 160341.36..160341.56 para el agregado de hash.
- 0.00..28.53 para el scan secuencial.
- 160459.08..160467.64 para el merge join.
- 116.89..121.02 para el sort.
- 160467.64..160467.64 para el hash.
- 160467.89..374251.77 para el join de hash.

Y el total fue:

409409.44..409413.57 del sort, que este fue el costo relativo del query completo.

- c. ¿Cuántos escaneos secuenciales reporta Postgres? ¿Cuáles son?
Reporta 2 escaneos secuenciales.
 - i. Escaneo secuencial para la tabla de aerolíneas.
 - ii. Escaneo secuencial para la tabla de vuelos.
- d. Conociendo los índices existentes en la base de datos y el execution plan, cree los índices necesarios para mejorar el tiempo de respuesta del query, de manera que se deje en evidencia las instrucciones de creación del índice.

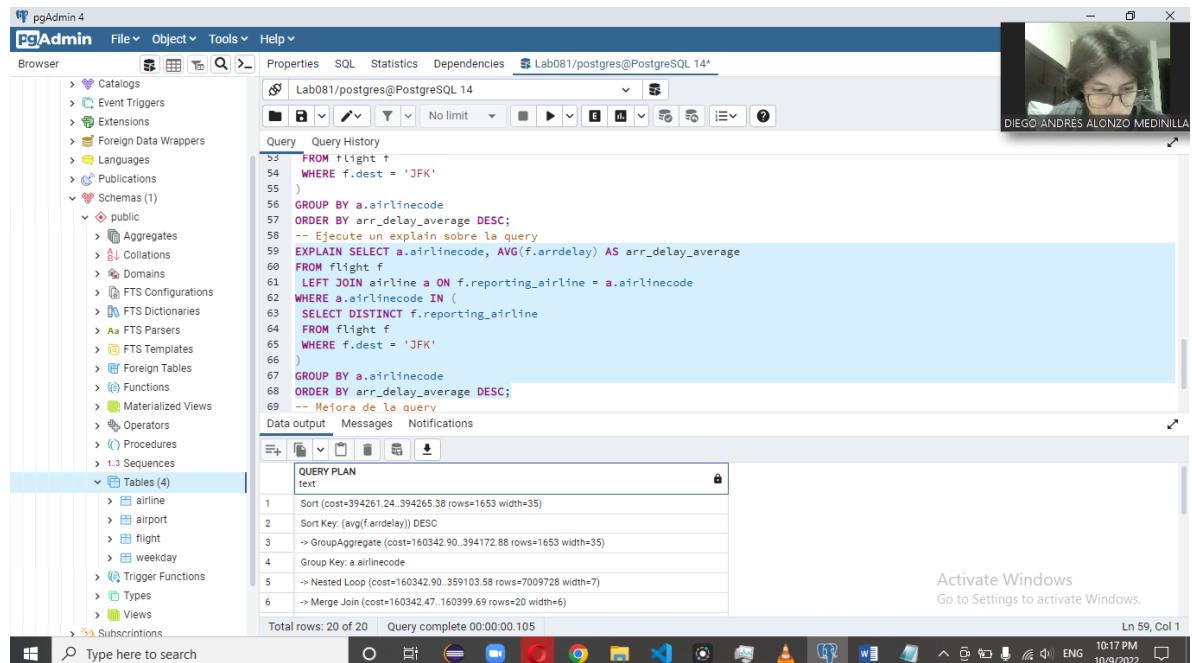
Crear un índice para el reporting airline, de manera que sea casi un group by.

```

71 create index on flight(reportng_airline)
72
Data output Messages Notifications
CREATE INDEX
Query returned successfully in 1 min 4 secs.
Total rows: 21 of 21 Query complete 00:01:04.548
Ln 71, Col 1

```

Para la creación de solo este índice mejoró y esta es la explicación.



```

pgAdmin 4
File Object Tools Help
Browser Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*
Query Query History
33   FROM flight f
34     WHERE f.dest = 'JFK'
35   )
36 GROUP BY a.airlinecode
37 ORDER BY arr_delay_average DESC;
38 -- Ejecutar un explain sobre la query
39 EXPLAIN SELECT a.airlinecode, AVG(f.arrdelay) AS arr_delay_average
40 FROM flight f
41   LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
42 WHERE a.airlinecode IN (
43   SELECT DISTINCT f.reporting_airline
44   FROM flight f
45     WHERE f.dest = 'JFK'
46   )
47 GROUP BY a.airlinecode
48 ORDER BY arr_delay_average DESC;
49 -- Mejora de la query
Data output Messages Notifications
QUERY PLAN
text
1  Sort (cost=394261.24..394265.38 rows=1653 width=35)
2  Sort Key: (avg(f.arrdelay)) DESC
3  -> GroupAggregate (cost=160342.90..394172.88 rows=1653
width=35)
4  Group Key: a.airlinecode
5  -> Nested Loop (cost=160342.90..359103.58 rows=7009728
width=7)
6  -> Merge Join (cost=160342.47..160399.69 rows=20
width=6)
Merge Cond: (a.airlinecode = f_1.reporting_airline)
-> Index Only Scan using airline_airlinecode_idx on airline
a (cost=0.28..53.07 rows=1653 width=3)
-> Sort (cost=160342.19..160342.24 rows=20 width=3)
Sort Key: f_1.reporting_airline
-> HashAggregate (cost=160341.36..160341.56
rows=20 width=3)
Group Key: f_1.reporting_airline
-> Gather (cost=2456.70..160014.82 rows=130615
width=3)
Workers Planned: 2
-> Parallel Bitmap Heap Scan on flight f_1
(cost=1456.70..145953.32 rows=54423 width=3)
Recheck Cond: (dest = 'JFK'::text)
-> Bitmap Index Scan on flight_dest_idx
(cost=0.00..1424.04 rows=130615 width=0)
Index Cond: (dest = 'JFK'::text)

```

Activate Windows
Go to Settings to activate Windows.

Ln 59, Col 1
10:17 PM 10/9/2022

"Sort (cost=394261.24..394265.38 rows=1653 width=35)"
" Sort Key: (avg(f.arrdelay)) DESC"
" -> GroupAggregate (cost=160342.90..394172.88 rows=1653 width=35)"
" Group Key: a.airlinecode"
" -> Nested Loop (cost=160342.90..359103.58 rows=7009728 width=7)"
" -> Merge Join (cost=160342.47..160399.69 rows=20 width=6)"
" Merge Cond: (a.airlinecode = f_1.reporting_airline)"
" -> Index Only Scan using airline_airlinecode_idx on airline a (cost=0.28..53.07 rows=1653 width=3)"
" -> Sort (cost=160342.19..160342.24 rows=20 width=3)"
" Sort Key: f_1.reporting_airline"
" -> HashAggregate (cost=160341.36..160341.56 rows=20 width=3)"
" Group Key: f_1.reporting_airline"
" -> Gather (cost=2456.70..160014.82 rows=130615 width=3)"
" Workers Planned: 2"
" -> Parallel Bitmap Heap Scan on flight f_1 (cost=1456.70..145953.32 rows=54423 width=3)"
" Recheck Cond: (dest = 'JFK'::text)"
" -> Bitmap Index Scan on flight_dest_idx (cost=0.00..1424.04 rows=130615 width=0)"
" Index Cond: (dest = 'JFK'::text)"

```

"
    -> Index Scan using flight_reporting_airline_idx on flight f
(cost=0.43..6430.33 rows=350486 width=7)
"
Index Cond: (reporting_airline = a.airlinecode)
"Sort (cost=394261.24..394265.38 rows=1653 width=35)"

```

Y se tardó 4.448 segundos en ejecutarse

The screenshot shows the pgAdmin 4 interface with a query window displaying the following SQL code:

```

53   FROM flight f
54   WHERE f.dest = 'JFK'
55   )
56   GROUP BY a.airlinecode
57   ORDER BY arr_delay_average DESC;
58   -- Ejecuta un explain sobre la query
59   EXPLAIN SELECT a.airlinecode, AVG(f.arrdelay) AS arr_delay_average
60   FROM flight f
61   LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
62   WHERE a.airlinecode IN (
63     SELECT DISTINCT f.reporting_airline
64     FROM flight f
65     WHERE f.dest = 'JFK'
66   )
67   GROUP BY a.airlinecode
68   ORDER BY arr_delay_average DESC;
69   -- Mejora de la query

```

The results pane shows a table with the following data:

	airlinecode	arr_delay_average
1	AA	12.6071904035713981
2	OH	11.81746768398001
3	YV	11.77518143360080
4	UA	11.29132216668018
5	B6	11.08418439051812
6	CO	10.97903728029130

Total rows: 11 of 11 | Query complete 00:00:04.448

Crear el Nuevo index de dest y reporting airline

The screenshot shows the pgAdmin 4 interface with a query window displaying the following SQL code:

```

60   FROM flight f
61   LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
62   WHERE a.airlinecode IN (
63     SELECT DISTINCT f.reporting_airline
64     FROM flight f
65     WHERE f.dest = 'JFK'
66   )
67   GROUP BY a.airlinecode
68   ORDER BY arr_delay_average DESC;
69   -- Mejora de la query
70   create index on airline(airlinecode)
71   create index on flight(reporting_airline)
72   create index on flight(reporting_airline, dest)
73   create index on flight(reporting_airline, airlinecode)
74   drop index flight_dest_reporting_airline_idx
75

```

The results pane shows the message:

CREATE INDEX

Query returned successfully in 1 min 7 secs.

Total rows: 20 of 20 | Query complete 00:01:07.292

Con dicho índice el tiempo de ejecución fue mayor, 5.044 segundos.

```

-- from flight f
  WHERE f.dest = 'JFK'
)
GROUP BY a.airlinecode
ORDER BY arr_delay_average DESC;
-- Ejecute un explain sobre la query
EXPLAIN SELECT a.airlinecode, AVG(f.arrdelay) AS arr_delay_average
FROM flight f
LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
WHERE a.airlinecode IN (
SELECT DISTINCT f.reporting_airline
FROM flight f
WHERE f.dest = 'JFK'
)
GROUP BY a.airlinecode
ORDER BY arr_delay_average DESC;
-- Mejora de la query
  
```

Total rows: 11 of 11 Query complete 00:00:05.044

"Sort (cost=252016.17..252020.30 rows=1653 width=35)"
" Sort Key: (avg(f.arrdelay)) DESC"
" -> Finalize GroupAggregate (cost=251504.89..251927.81 rows=1653 width=35)"
" Group Key: a.airlinecode"
" -> Gather Merge (cost=251504.89..251890.61 rows=3306 width=35)"
" Workers Planned: 2"
" -> Sort (cost=250504.86..250509.00 rows=1653 width=35)"
" Sort Key: a.airlinecode"
" -> Partial HashAggregate (cost=250399.97..250416.50 rows=1653 width=35)"
" Group Key: a.airlinecode"
" -> Hash Join (cost=78731.09..235796.37 rows=2920720 width=7)"
" Hash Cond: (f.reporting_airline = a.airlinecode)"
" -> Parallel Seq Scan on flight f (cost=0.00..145759.20 rows=2920720 width=7)"
" -> Hash (cost=78730.84..78730.84 rows=20 width=6)"
" -> Nested Loop (cost=0.43..78730.84 rows=20 width=6)"
" Join Filter: (a.airlinecode = f_1.reporting_airline)"

```

"
-> Unique (cost=0.43..78202.08 rows=20
width=3)"
"
-> Index Only Scan using
flight_reporting_airline_dest_idx on flight f_1 (cost=0.43..77875.54
rows=130615 width=3)"
"
Index Cond: (dest = 'JFK'::text)"
"
-> Materialize (cost=0.00..36.80 rows=1653
width=3)"
"
-> Seq Scan on airline a
(cost=0.00..28.53 rows=1653 width=3)"
"Sort (cost=252016.17..252020.30 rows=1653 width=35)"

```

Es debido a que tiene el problema que al final termina generando un escaneo secuencial.

- e. Costo relativo total es de 394261.24..394265.38

```

"Sort (cost=394261.24..394265.38 rows=1653 width=35)"
" Sort Key: (avg(f.arrdelay)) DESC"
" -> GroupAggregate (cost=160342.90..394172.88 rows=1653
width=35)"
"
Group Key: a.airlinecode"
"
-> Nested Loop (cost=160342.90..359103.58 rows=7009728
width=7)"
"
-> Merge Join (cost=160342.47..160399.69 rows=20
width=6)"
"
Merge Cond: (a.airlinecode = f_1.reporting_airline)"
"
-> Index Only Scan using airline_airlinecode_idx on airline
a (cost=0.28..53.07 rows=1653 width=3)"
"
-> Sort (cost=160342.19..160342.24 rows=20 width=3)"
"
Sort Key: f_1.reporting_airline"
"
-> HashAggregate (cost=160341.36..160341.56
rows=20 width=3)"
"
Group Key: f_1.reporting_airline"
"
-> Gather (cost=2456.70..160014.82 rows=130615
width=3)"
"
Workers Planned: 2"
"
-> Parallel Bitmap Heap Scan on flight f_1
(cost=1456.70..145953.32 rows=54423 width=3)"
"
Recheck Cond: (dest = 'JFK'::text)"
"
-> Bitmap Index Scan on flight_dest_idx
(cost=0.00..1424.04 rows=130615 width=0)"
"
Index Cond: (dest = 'JFK'::text)"
"
-> Index Scan using flight_reporting_airline_idx on flight f
(cost=0.43..6430.33 rows=350486 width=7)"
"
Index Cond: (reporting_airline = a.airlinecode)"

```

```

-- from flight f
  WHERE f.dest = 'JFK'
)
GROUP BY a.airlinecode
ORDER BY arr_delay_average DESC;
-- Ejecutar un explain sobre la query
EXPLAIN SELECT a.airlinecode, AVG(f.arrdelay) AS arr_delay_average
FROM flight f
LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
WHERE a.airlinecode IN (
  SELECT DISTINCT f.reporting_airline
  FROM flight f
  WHERE f.dest = 'JFK'
)
GROUP BY a.airlinecode
ORDER BY arr_delay_average DESC;
-- Mejora de la query
  
```

Activate Windows
Go to Settings to activate Windows.

Y el tiempo de ejecución fue 4.493 segundos.

```

-- from flight f
  WHERE f.dest = 'JFK'
)
GROUP BY a.airlinecode
ORDER BY arr_delay_average DESC;
-- Ejecutar un explain sobre la query
EXPLAIN SELECT a.airlinecode, AVG(f.arrdelay) AS arr_delay_average
FROM flight f
LEFT JOIN airline a ON f.reporting_airline = a.airlinecode
WHERE a.airlinecode IN (
  SELECT DISTINCT f.reporting_airline
  FROM flight f
  WHERE f.dest = 'JFK'
)
GROUP BY a.airlinecode
ORDER BY arr_delay_average DESC;
-- Mejora de la query
  
```

airlinecode	arr_delay_average
AA	12.607194035713981
OH	11.8174676839800
YV	11.77518143360080
UA	11.29132218668018
B6	11.0841439051812
CO	10.97903728029130

Activate Windows
Go to Settings to activate Windows.

4. Ejercicio 4 – Consultas SARGABLES.

- Esta consulta devuelve los estudiantes asignados por curso en el año 2022, y posee un índice en la fecha de asignación, pero no es utilizado
 - Query original:

```

SELECT id_curso, COUNT(*) FROM asignacion
WHERE EXTRACT(YEAR, fecha_asignacion) = 2022
GROUP BY id_curso
  
```

- ii.

```
SELECT id_curso, COUNT(*) FROM asignacion
WHERE fecha_asignación>'31-12-2021' and fecha_asignacion<'01-12-2023'
GROUP BY id_curso
```
 - b. Esta consulta devuelve los cursos que los estudiantes con apellido "Alvarado" se asignaron, y posee un índice en los apellidos, pero no es utilizado
 - i.

```
SELECT id_curso, nombre_curso FROM asignacion
INNER JOIN
estudiante on asignacion.carnet = estudiante.carnet
WHERE SUBSTRING(apellidos, 0, 8) = 'Alvarado'
```
 - ii.

```
SELECT id_curso, nombre_curso FROM asignacion
INNER JOIN
estudiante on asignacion.carnet = estudiante.carnet
WHERE estudiante.apellidos = 'Alvarado'
```
 - c. Esta consulta devuelve las asignaciones cuyo tiempo total fueron más de 300 segundos. La columna tiempo_total almacena minutos y posee un índice que no es utilizado
 - i.

```
SELECT id_asignacion, tiempo_total FROM asignacion
WHERE tiempo_total*60 >= 300
```
 - ii.

```
SELECT id_asignacion, tiempo_total FROM asignacion
WHERE tiempo_total >= 5
```
5. Ejercicio 5 – Creación de triggers.

a. Creación de tablas

```

64 FROM flight f
65 WHERE f.dest = 'JFK'
66 )
67 GROUP BY a.airlinecode
68 ORDER BY arr_delay_average DESC;
69 -- Mejora de la query
70 create index on airline(airlinecode)
71 create index on flight(reporting_airline)
72 create index on flight(reporting_airline, dest)
73 create index on flight(reporting_airline, airlinecode)
74 drop index flight_arrdelay_idx
75 create index on flight(arrdelay)
76 -- Mejorar queries para no usar indices de funciones
77 Create table ASIGNACION(carnet VARCHAR(12), codigo_curso VARCHAR(6), fecha_asignacion DATE)
78 ASIGNACION_HISTORICA(carnet VARCHAR(12), codigo_curso VARCHAR(6), fecha_asignacion
79 DATE, fecha_eliminacion DATE)

```

Activate Windows
Go to Settings to activate Windows.

Ln 77, Col 1

Activate Windows
Go to Settings to activate Windows.

Ln 79, Col 30

- b. Crear un trigger que impida que una materia tenga más de 5 asignaciones para un mismo semestre.**

pgAdmin 4

File Object Tools Help

Browser

Servers (1)

- PostgreSQL 14
- Databases (2)
 - Lab081
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
- Schemas (1)
 - public
- Tables (4)
 - airline
 - airport
 - flight

Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*

Query History

```

15  ONREC, RECOMPLETION_DATE
16
17  drop table asignacion
18  -- Trigger mas de 5 asignaciones
19  create function asignacion_validation()
20  returns trigger as $$
21  declare asignacion_total_course int;
22
23  begin
24    select count(carnet) into asignacion_total_course
25    from asignacion where codigo_curso = new.codigo_curso;
26    if (asignacion_total_course > 4) then raise exception
27      'Las asignaciones del curso ya tienen su cupo maximo, no se puede asignar'
28      using errcode = '28088';
29    end if;
30    return new;
31  end; $$ language PLPGSQL
32
33  create trigger trigger_validation_asignation
34  before insert on asignacion
35  
```

Data output Messages Notifications

CREATE FUNCTION

Query returned successfully in 254 msec.

Total rows: 2 of 2 Query complete 00:00:00.254

Type here to search

Activate Windows

Query returned successfully in 254 msec. Ln 97, Col 26

10/10/2022 1:24 AM ENG

Crear la función.

pgAdmin 4

File Object Tools Help

Browser

Servers (1)

- PostgreSQL 14
- Databases (2)
 - Lab081
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Tables (4)
 - airline
 - airport
 - flight

Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*

Query History

```

15  select count(carnet) into asignacion_total_course
16  from asignacion where codigo_curso = new.codigo_curso;
17  if (asignacion_total_course > 4) then raise exception
18    'Las asignaciones del curso ya tienen su cupo maximo, no se puede asignar'
19    using errcode = '28088';
20  end if;
21  return new;
22 end; $$ language PLPGSQL
23
24  create trigger trigger_validation_asignation
25  before insert on asignacion
26  for each row execute procedure asignacion_validation()
27
28  drop function asignacion_validation
29  drop trigger trigger_validation_asignation on asignacion
30
31  Drop table asignacion
32
33  
```

Data output Messages Notifications

CREATE TRIGGER

Query returned successfully in 88 msec.

Total rows: 2 of 2 Query complete 00:00:00.088

Type here to search

Activate Windows

Query returned successfully in 88 msec. Ln 97, Col 1

10/10/2022 1:24 AM ENG

Crear el trigger.

pgAdmin 4

File Object Tools Help

Browser Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*

Servers (1) PostgreSQL 14 Databases (2) Lab081

- Casts
- Catalogs
- Event Triggers
- Extensions
- Foreign Data Wrappers
- Languages
- Publications
- Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - 1.3 Sequences
- Tables (4)
 - airline
 - airport
 - flight

Query Query History

```

86    SELECT COUNT(*) AS total FROM asignacion WHERE codigo_curso = new.codigo_curso;
87    IF (asignacion.total_course > 4) THEN RAISE EXCEPTION
88      'Las asignaciones del curso ya tienen su cupo maximo, no se puede asignar'
89      USING ERRORE = '200808';
90    END IF;
91    RETURN NEW;
92 END; $$ LANGUAGE PLPGSQL;
93
94 CREATE TRIGGER trigger_validation_asignacion
95 BEFORE INSERT ON asignacion
96 FOR EACH ROW EXECUTE PROCEDURE asignacion_validation();
97
98 DROP FUNCTION asignacion_validation;
99 DROP TRIGGER trigger_validation_asignacion ON asignacion;
100
101 DROP TABLE asignacion;
```

Data output Messages Notifications

CREATE TRIGGER

Query returned successfully in 88 msec.

Total rows: 2 of 2 Query complete 00:00:00.088

Activate Windows

Query returned successfully in 88 msec. Ln 97, Col 1

10:24 AM 10/10/2022

Insertar los valores

pgAdmin 4

File Object Tools Help

Browser Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*

Servers (1) PostgreSQL 14 Databases (2) Lab081

- Casts
- Catalogs
- Event Triggers
- Extensions
- Foreign Data Wrappers
- Languages
- Publications
- Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - 1.3 Sequences
- Tables (4)
 - airline
 - airport
 - flight

Query Query History

```

98 drop function asignacion_validation;
99 drop trigger trigger_validation_asignacion on asignacion;
100
101 drop table asignacion;
102 drop table asignacionHistorica;
103
104 insert into asignacion(carnet, codigo_curso, fecha_asignacion)
105 values(489320483012, 328291, '01-01-2022'),
106 (489320483012, 328291, '01-11-2022'),
107 (789320483012, 328291, '10-01-2022'),
108 (843920439023, 328291, '03-01-1992'),
109 (972837429, 328291, '03-12-2021');
110 insert into asignacion(carnet, codigo_curso, fecha_asignacion)
111 values(8439248932, 38291, '05-07-2008');
112
113 select count(carnet), codigo_curso from asignacion group by codigo_curso;
```

Data output Messages Notifications

INSERT 0 5

Query returned successfully in 211 msec.

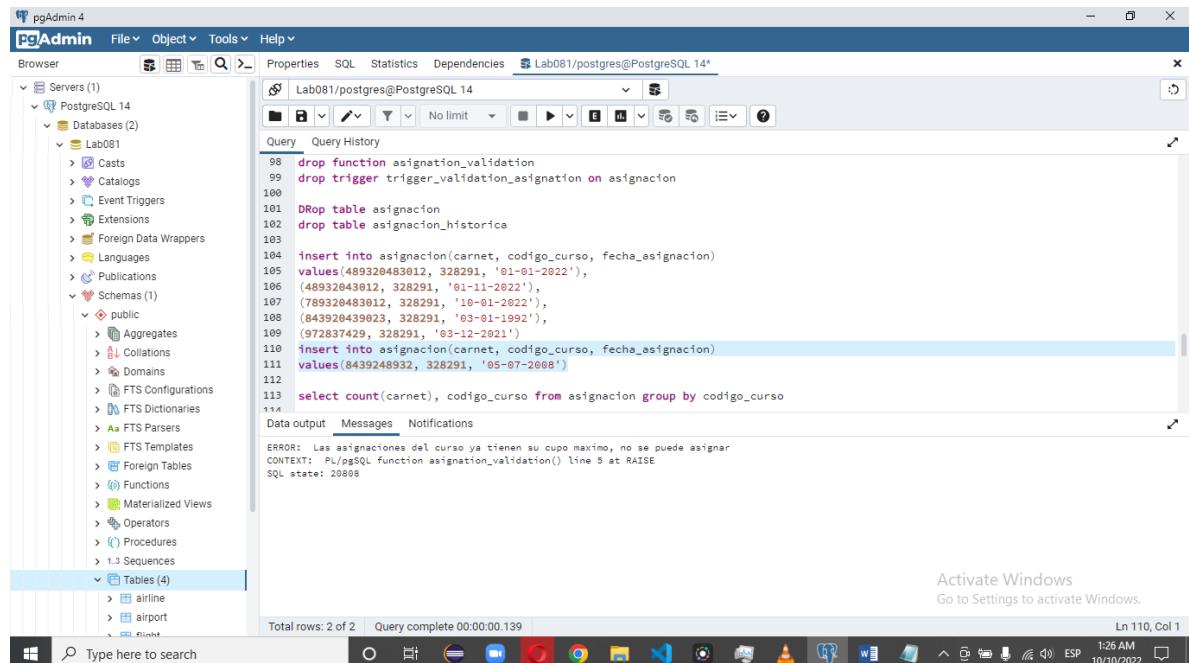
Total rows: 2 of 2 Query complete 00:00:00.211

Activate Windows

Query returned successfully in 211 msec. Ln 109, Col 34

10:25 AM 10/10/2022

Prueba de que funcionó el trigger.



The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays the database structure under 'Lab081'. In the main 'Query' pane, a SQL script is being run:

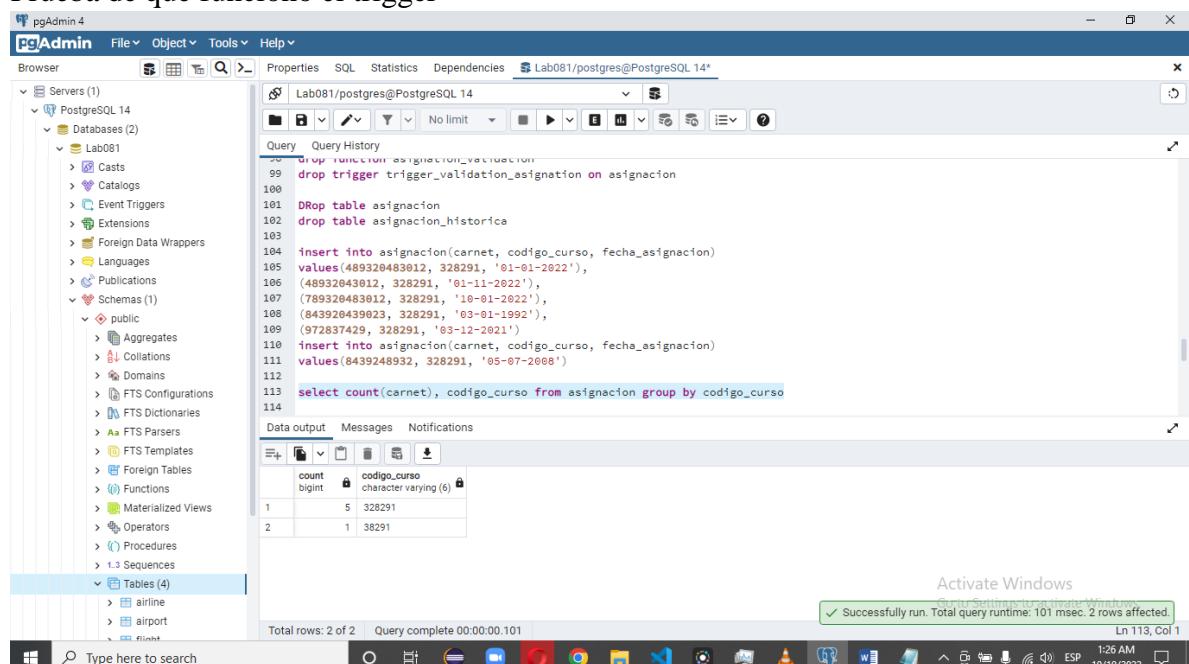
```

98 drop function asignacion_validation
99 drop trigger trigger_validation_asignacion on asignacion
100
101 DRop table asignacion
102 drop table asignacion_historica
103
104 insert into asignacion(carnet, codigo_curso, fecha_asignacion)
105 values(489320483012, 328291, '01-01-2022'),
106 (489320483012, 328291, '01-11-2022'),
107 (789320483012, 328291, '10-01-2022'),
108 (843920439023, 328291, '03-01-1992'),
109 (972837429, 328291, '03-12-2021')
110 insert into asignacion(carnet, codigo_curso, fecha_asignacion)
111 values(8439248932, 328291, '05-07-2008')
112
113 select count(carnet), codigo_curso from asignacion group by codigo_curso
114

```

An error message is visible: "ERROR: Las asignaciones del curso ya tienen su cupo maximo, no se puede asignar". The status bar at the bottom indicates "Query complete 00:00:00.139".

Prueba de que funciono el trigger



The screenshot shows the pgAdmin 4 interface. The 'Query' pane contains the same SQL script as the previous screenshot. The status bar at the bottom indicates "Query complete 00:00:00.101". A message bar at the bottom right says "Successfully run. Total query runtime: 101 msec. 2 rows affected." The status bar also shows "Ln 113, Col 1".

count	codigo_curso
1	5 328291
2	1 328291

- Crear un trigger que registre en la tabla histórica de asignaciones las asignaciones que son eliminadas.

```
140 FOR EACH ROW
141     EXECUTE PROCEDURE VALIDACION_ASIGNACION();
142 -- Trigger que ingrese en la tabla histórica los eliminados de la tabla de asignacion
143 create function insert_historic()
144 returns trigger as $$
145 begin
146     insert into asignacion_histórica(carnet, codigo_curso, fecha_asignacion, fecha_eliminacion)
147     values (old.carnet, old.codigo_curso, old.fecha_asignacion, old.fecha_asignacion);
148     return old;
149 end; $$ language PLPGSQL;
150
151 drop function insert_historic()
152 drop trigger trigger_new_historic on asignacion
153
154 create trigger trigger_new_historic
155 before delete
156
157 ON asignacion
158 FOR EACH ROW
159     EXECUTE PROCEDURE insert_historic();
160
161 delete from asignacion where fecha_asignacion = '05-07-2008'
162 drop table asignacion
163 drop table asignacion_histórica
164 select * from asignacion
165
166 activate windows
```

```
149 end; $$ language PLPGSQL
150
151 drop function insert_historic()
152 drop trigger trigger_new_historic on asignacion
153
154 create trigger trigger_new_historic
155 before delete
156 ON asignacion
157 FOR EACH ROW
158     EXECUTE PROCEDURE insert_historic();
159
160 delete from asignacion where fecha_asignacion = '05-07-2008'
161 drop table asignacion
162 drop table asignacion_historica
163 select * from asignacion
164 select* from asignacion_historica
Data output Messages Notifications
DELETE 1
Query returned successfully in 87 msec.
```

Total rows: 2 of 2 Query complete 00:00:00.087

Activate Windows
C:\Settings\Temp\trans_WinAuth
Query returned successfully in 87 msec.
Ln 159, Col 5

1:14 PM 10/10/2022


```
149 end; $$ language PLPGSQL
150
151 drop function insert_historic()
152 drop trigger trigger_new_historic on asignacion
153
154 create trigger trigger_new_historic
155 before delete
156 ON asignacion
157 FOR EACH ROW
158     EXECUTE PROCEDURE insert_historic();
159
160 delete from asignacion where fecha_asignacion = '05-07-2008'
161 drop table asignacion
162 drop table asignacion_historica
163 select * from asignacion
164 select* from asignacion_historica
Data output Messages Notifications
DELETE 1
Query returned successfully in 87 msec.
```

Total rows: 2 of 2 Query complete 00:00:00.087

Activate Windows
C:\Settings\Temp\trans_WinAuth
Query returned successfully in 87 msec.
Ln 159, Col 5

1:14 PM 10/10/2022

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of the database structure under 'PostgreSQL 14'. The 'Tables' node is currently selected, showing four tables: 'airline', 'city', 'flight', and 'airport'. The 'Query' pane on the right contains a block of PL/SQL code. The code includes several SQL statements: dropping a function named 'insert_historic', dropping a trigger named 'trigger_new_historic' on the 'asignacion' table, creating a new trigger 'trigger_new_historic' on the 'asignacion' table that fires before a delete operation, and executing the 'insert_historic' procedure. It also includes a delete statement for rows where 'fecha_asignacion' is '05-07-2008', dropping the 'asignacion' table, and creating a new table 'asignacionHistorica' with the same structure. Finally, it performs a select operation on the original 'asignacion' table and then on the newly created 'asignacionHistorica' table. Below the code, a data grid shows a single row from the 'asignacion' table. The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.107'. The system tray at the bottom right shows the date and time as '10/10/2022 1:14 PM'.

```
149 end; $$ language PLPGSQL
150
151 drop function insert_historic()
152 drop trigger trigger_new_historic on asignacion
153
154 create trigger trigger_new_historic
155 before delete
156 ON asignacion
157 FOR EACH ROW
158     EXECUTE PROCEDURE insert_historic();
159
160 delete from asignacion where fecha_asignacion = '05-07-2008'
161 drop table asignacion
162 drop table asignacionHistorica
163 select * from asignacion
164 select* from asignacionHistorica
```

camet	codigo_curso	fecha_asignacion	fecha_eliminacion
1	8439248932	328291	2008-05-07