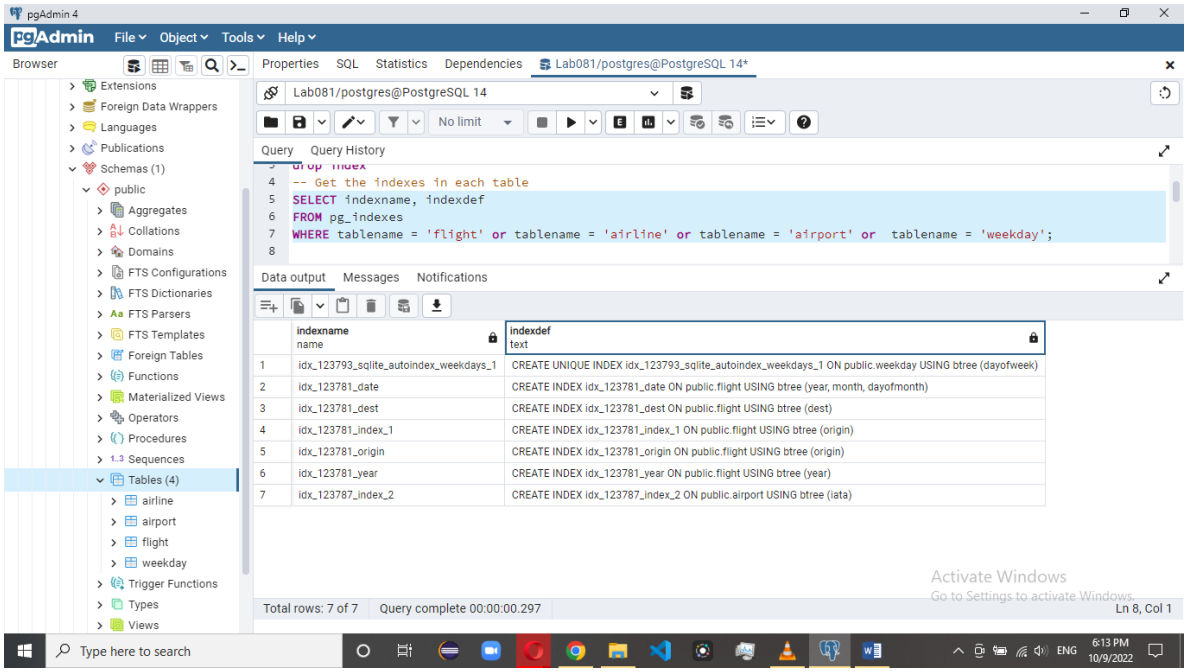
Laboratorio 08

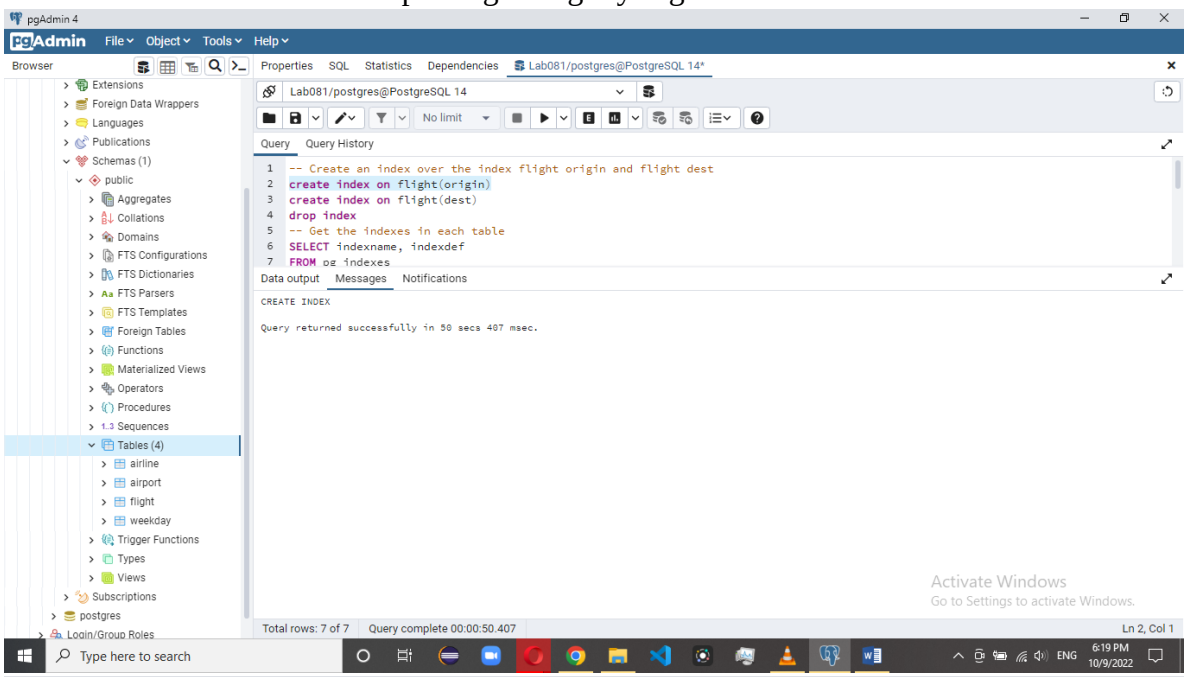
1. Ejercicio 1 – selectividad de los atributos.

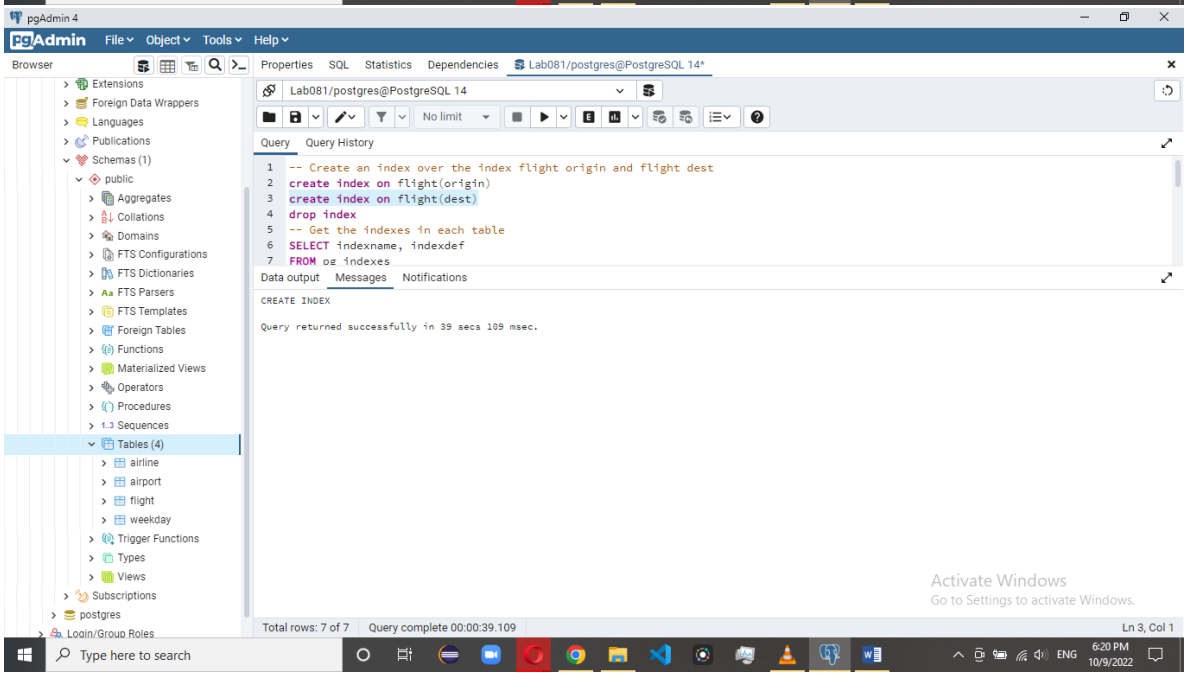
USUARIO	DETALLE
ID de usuario	ID del usuario
Número de serie de la báscula	Fecha y hora de cada pesaje
Fecha de nacimiento	Valor en libras del peso registrado.
Fecha del inicio de uso de servicio	
Altura	
Peso inicial al momento de uso del servicio.	
País de nacimiento	

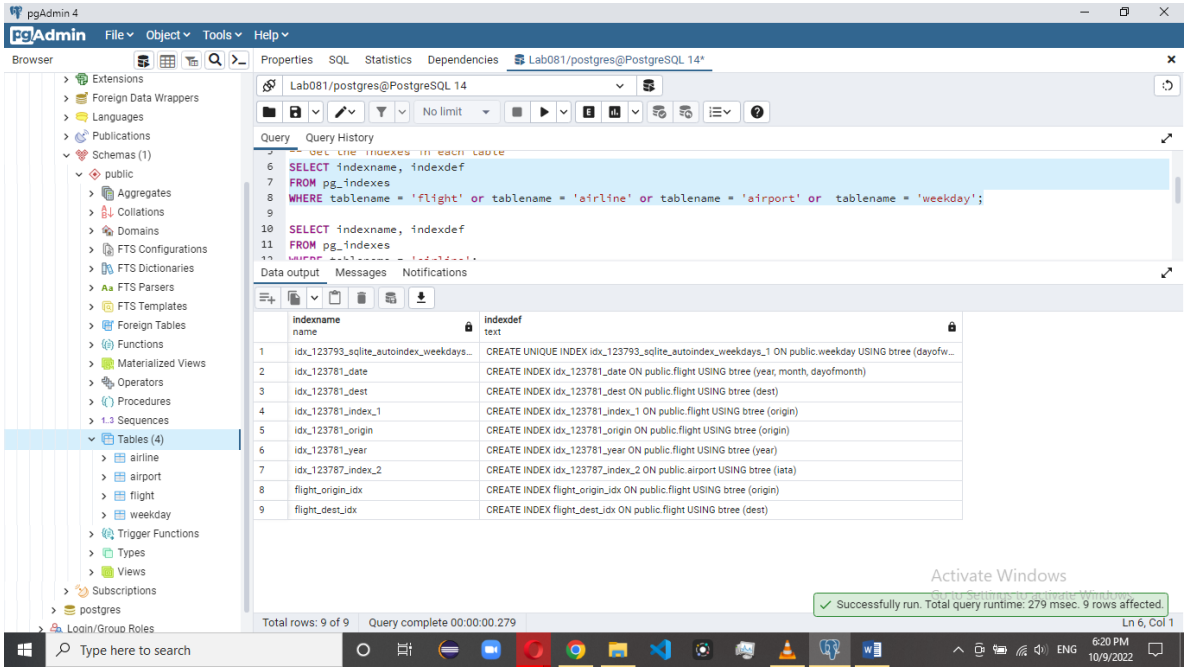
2. Ejercicio 2 – Declaración de índices.
   a. Investigue cómo enumerar los índices existentes en la base de datos y muéstrelos a continuación.



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

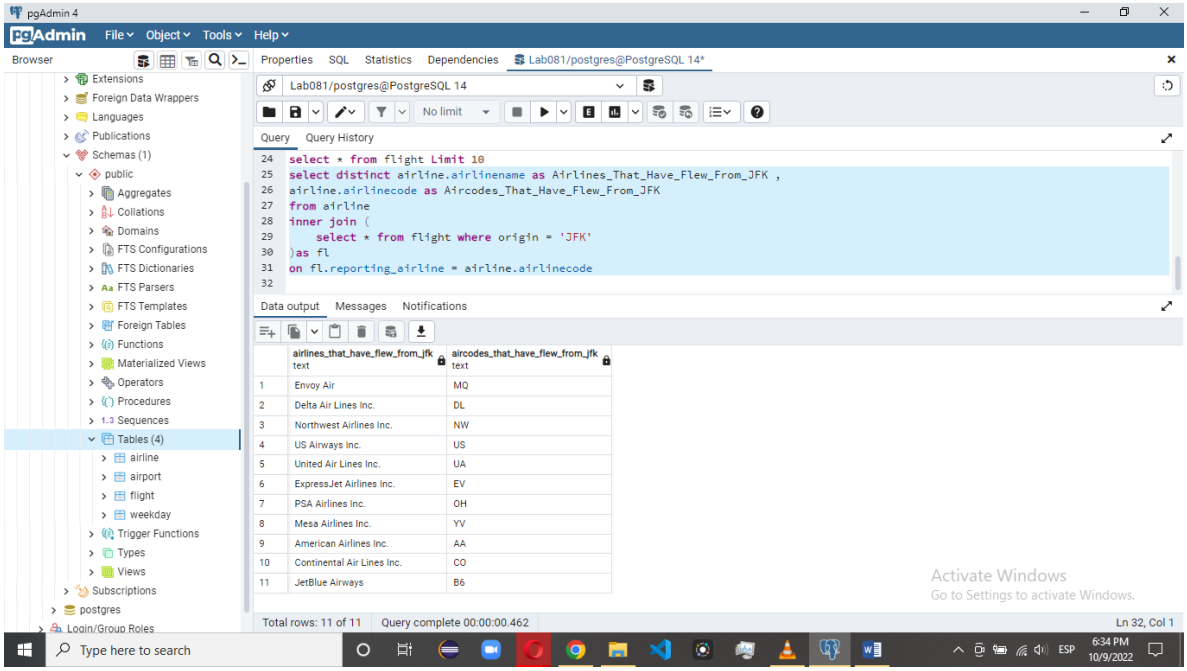




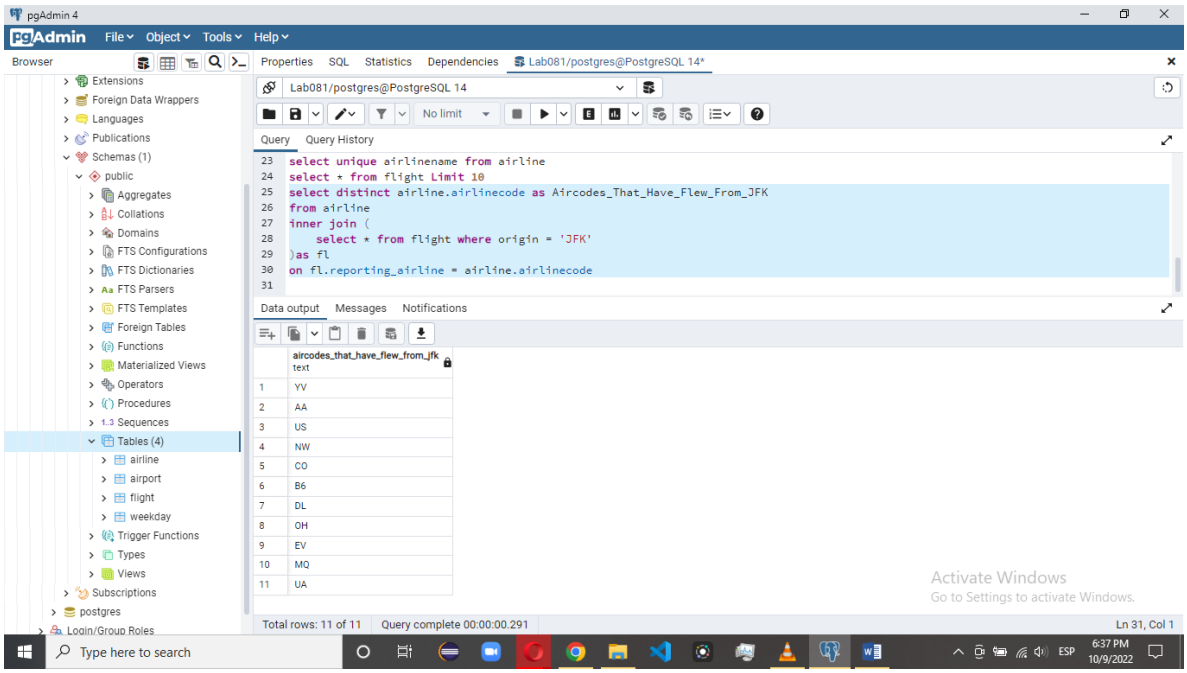


Como pueden ver se crearon esos índices.

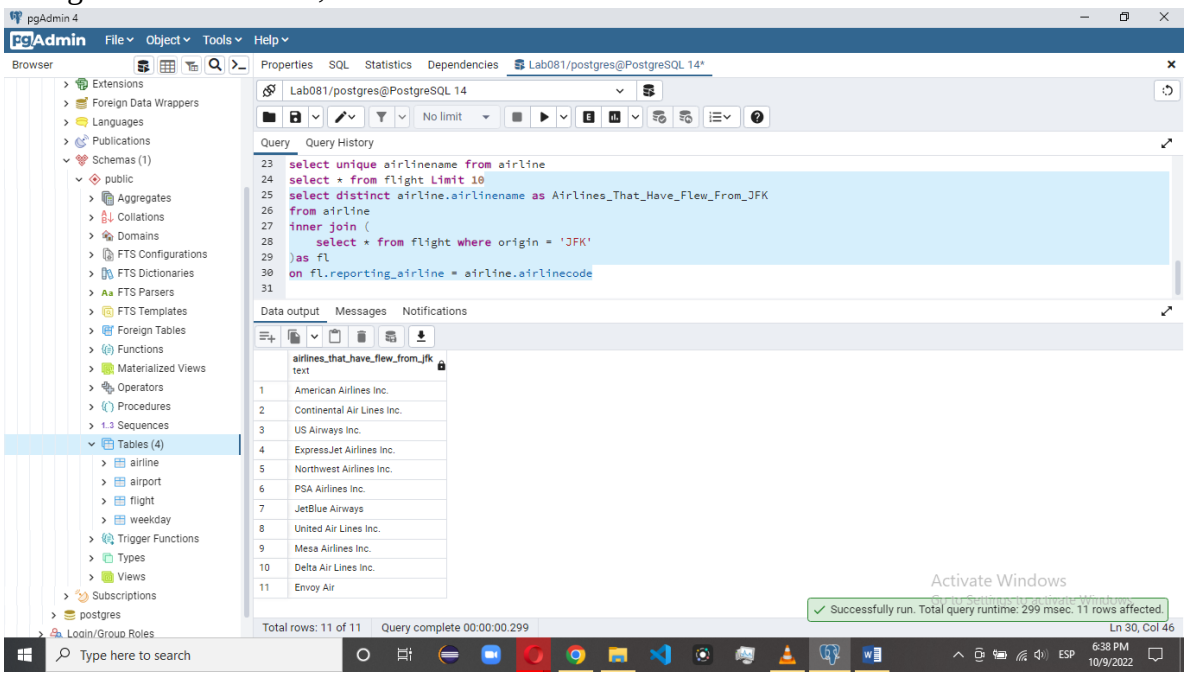
c. Query con los códigos de las aerolíneas que realizan vuelos que salen del aeropuerto JFK (sin duplicados), e indique cuanto tiempo toma la ejecución de su query.



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

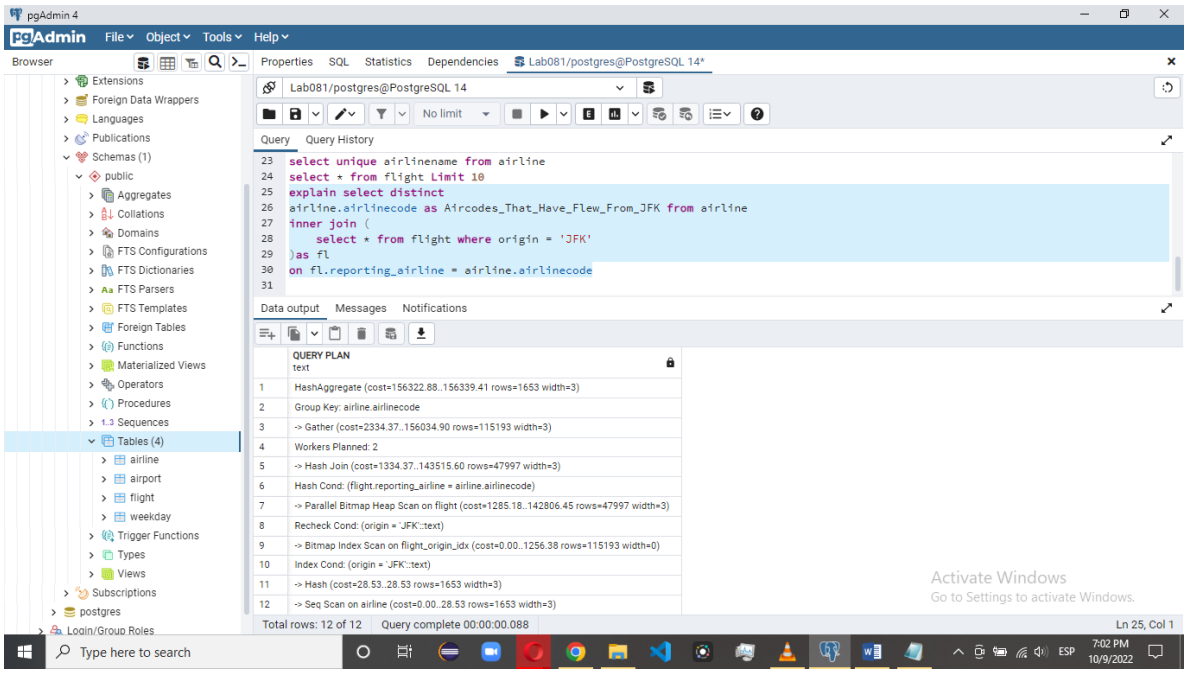


Código de las aerolíneas, se tardó 291 ms.



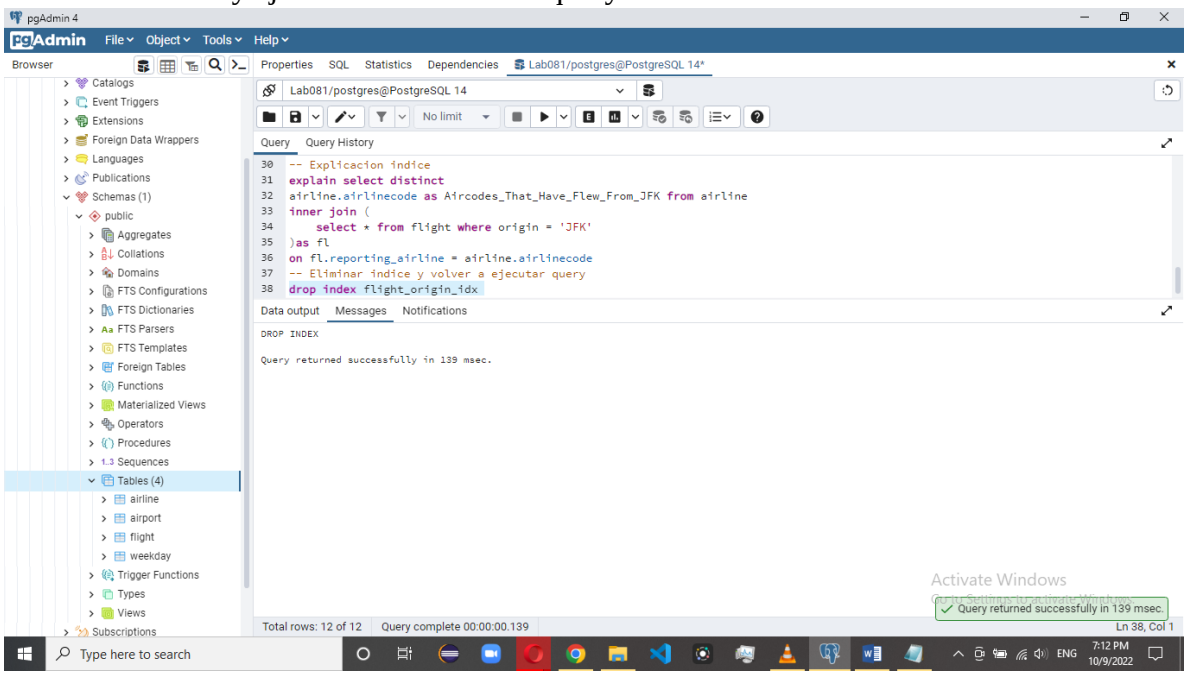
Nombre de las aerolíneas, se tardó 299 ms.

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

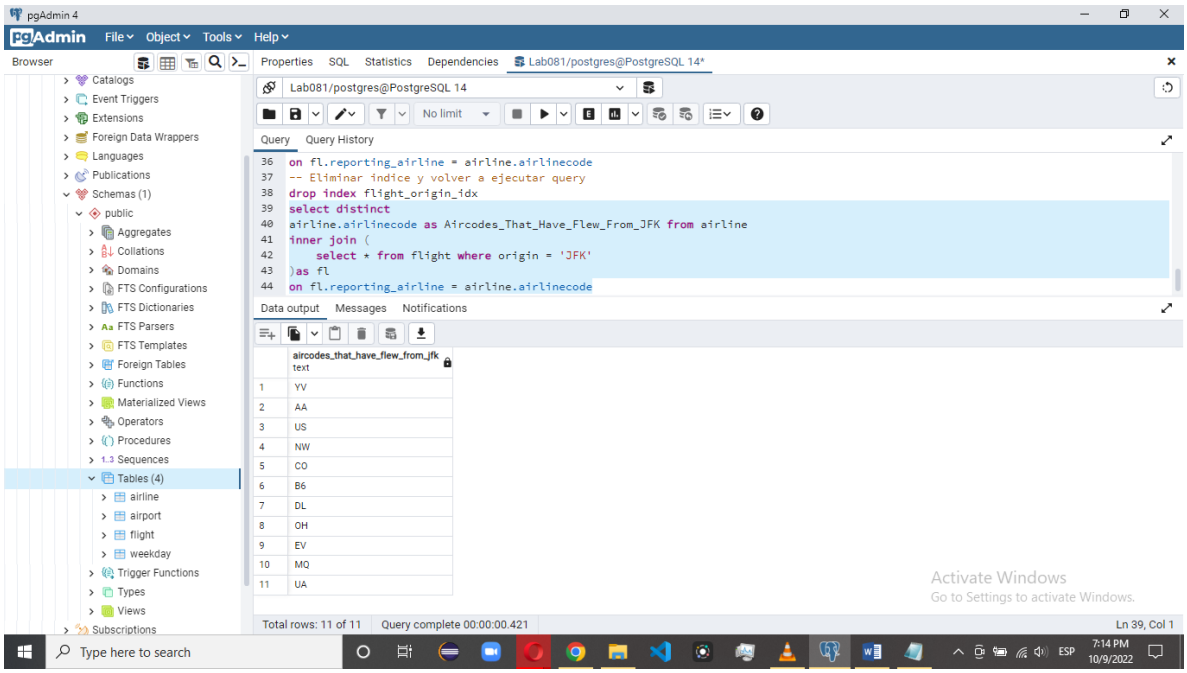


Se utiliza el índice flight_origin_idx para optimizar la búsqueda.

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

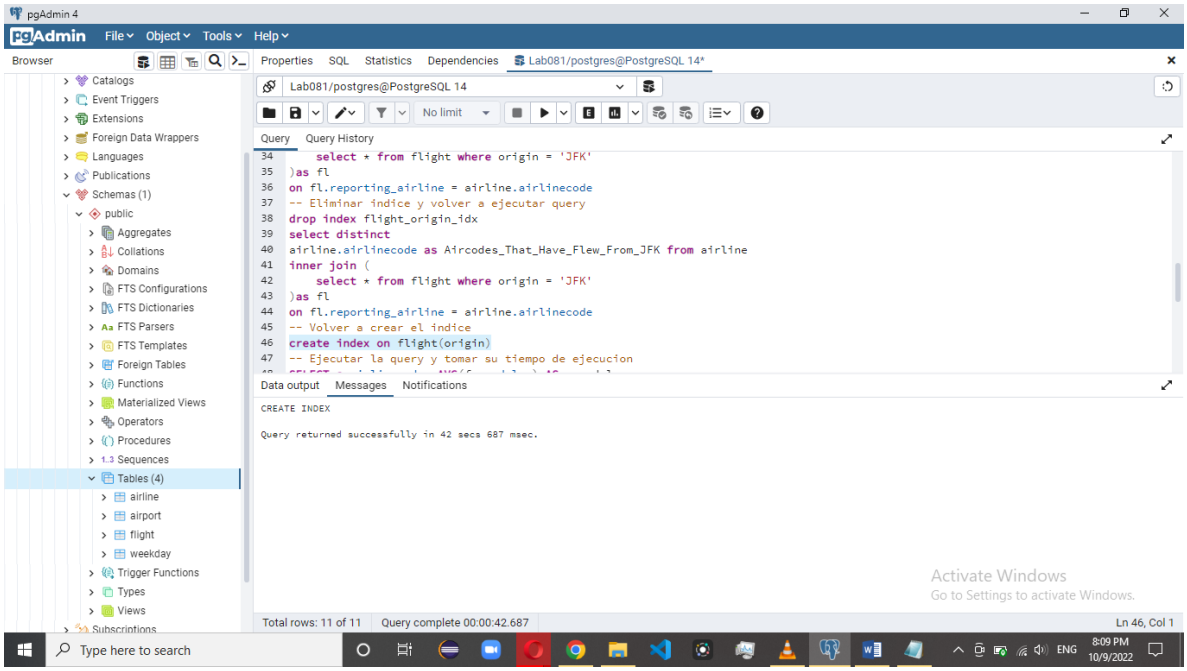


Eliminación del índice



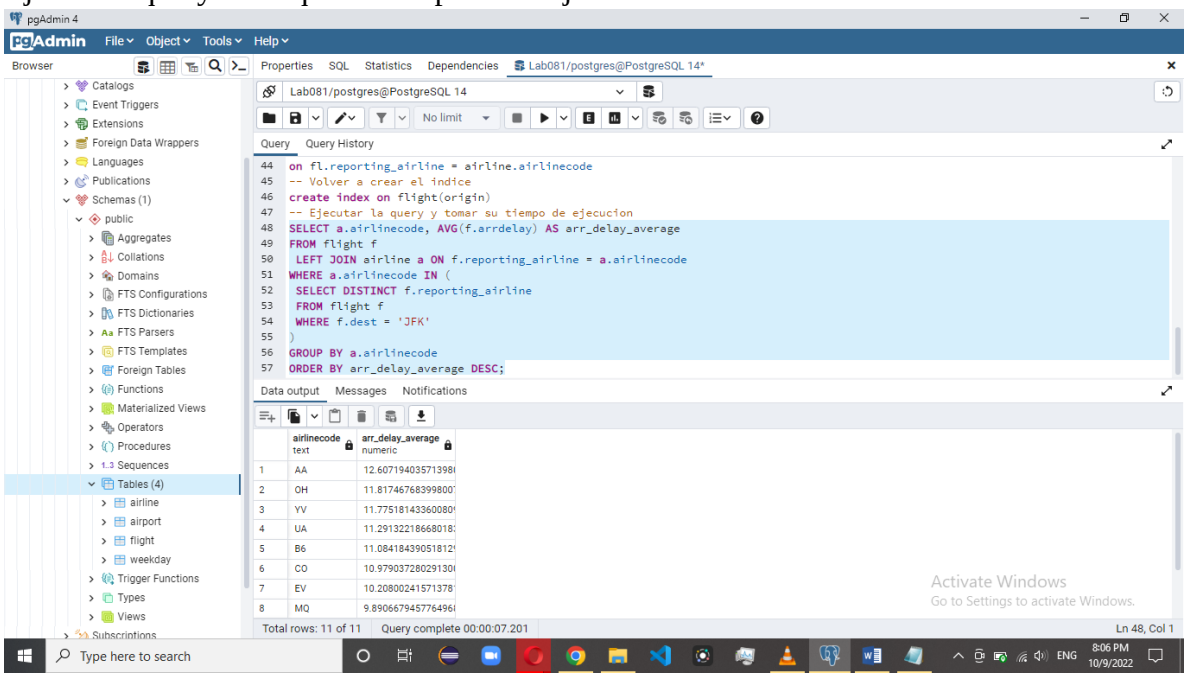
Ejecución de la query nuevamente, se ejecutó en 421 ms, casi lo que se tarda en hacer para dos columnas.

f. Cree nuevamente el indice eliminado.



3. Ejercicio 3 – Selección de índices.

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



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:

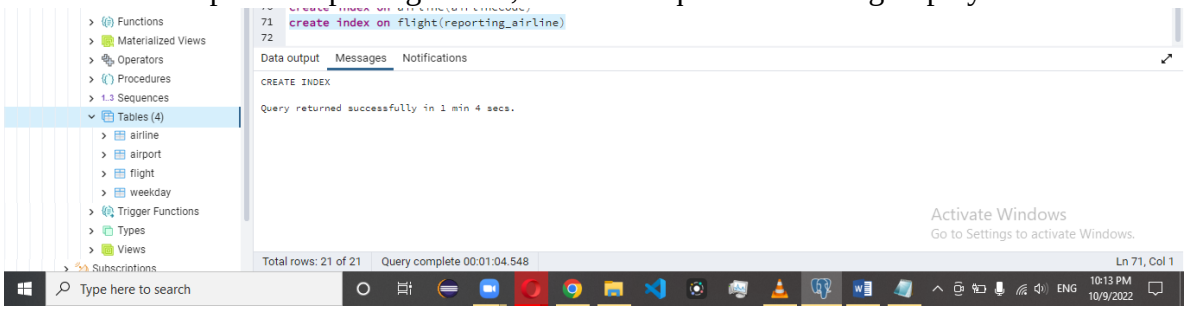
- 0.00..1424.04 para el bitmap index scan.
- 1456.70..145953.32 para el bitmap heap scan paralelo.
- 2456.70..160014.82 para la recolección.
- 160341.36..160341.56 para el agregado de hash.
- 0.00..28.53 para el scan secuencial.
- 160459.08..160467.64 para el merge join.
- 116.89..121.02 para el sort.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

Y el total fue:

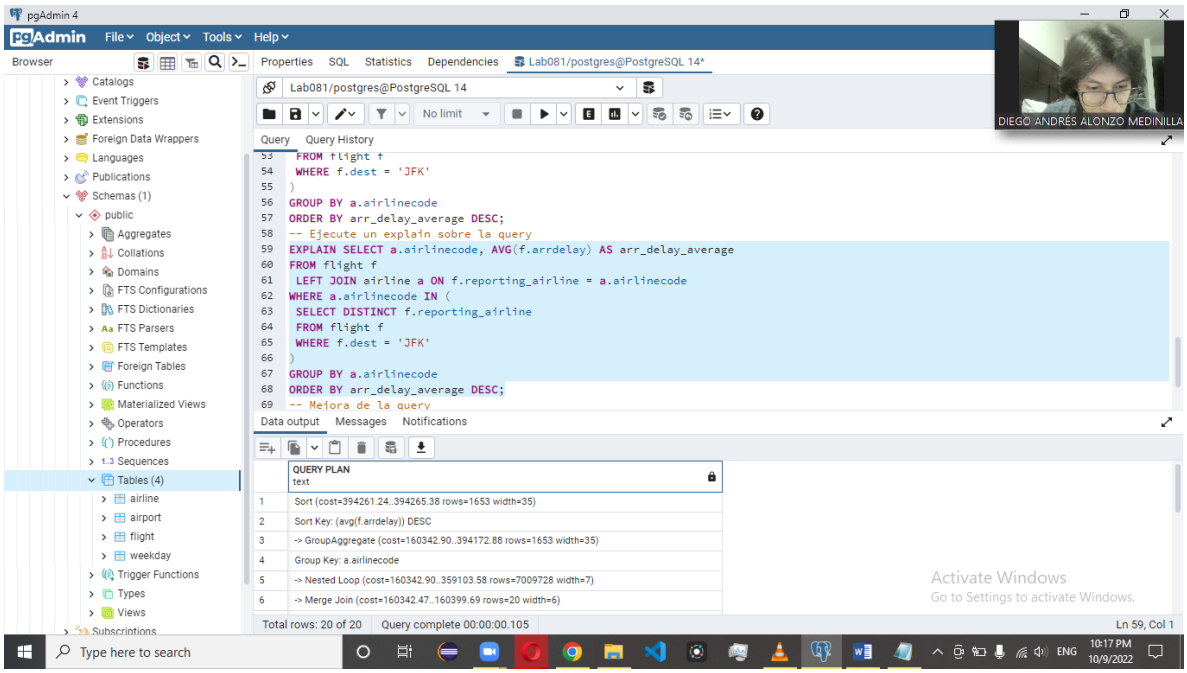
409409.44..409413.57 del sort, que este fue el costo relativo del query completo.

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.



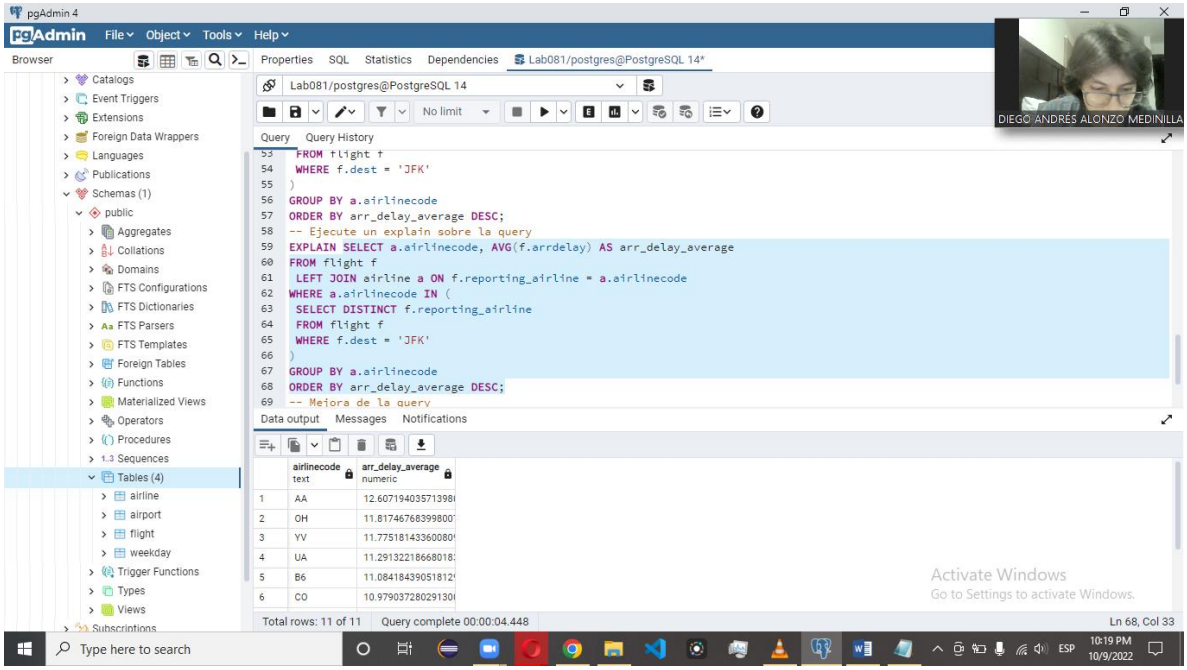
Para la creación de solo este índice mejoro y esta es la explicación.



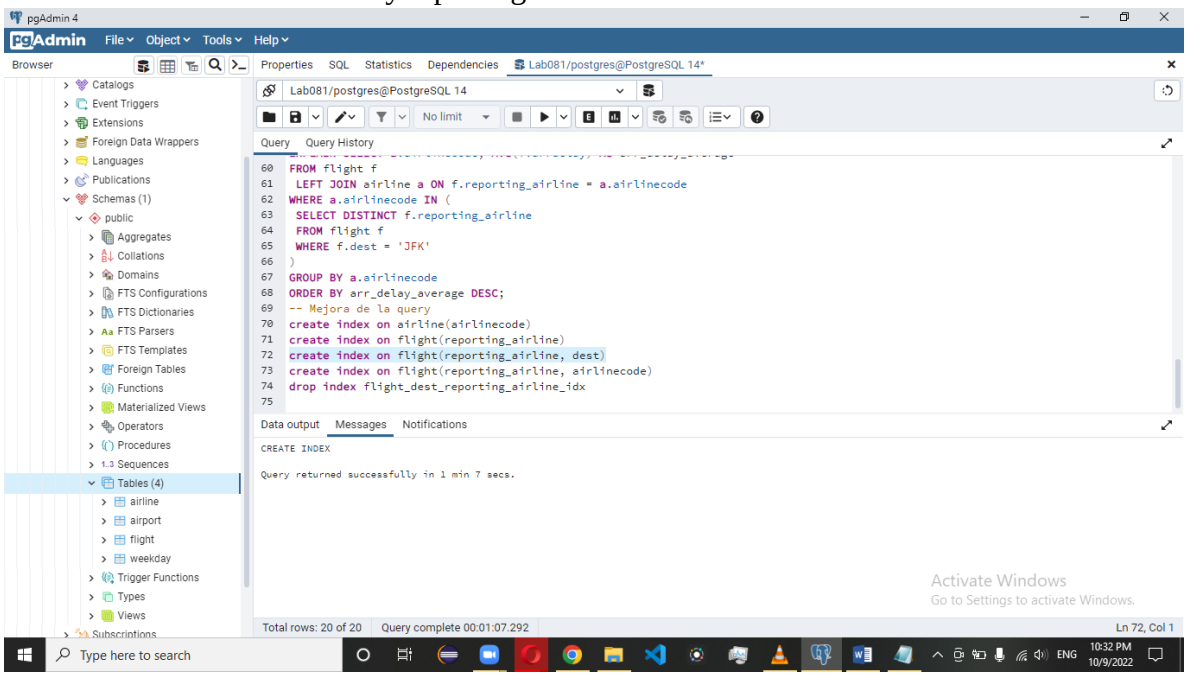
"Sort (cost=394261.24..394265.38 rows=1653 width=35)"
" Sort Key: (avg(f.arrdelay)) DESC"
" -> GroupAggregate (cost=160342.90..394172.88 rows=1653 width=35)"
" Group Key: a.airlinecode"
" -> Nested Loop (cost=160342.90..359103.58 rows=7009728 width=7)"
" -> Merge Join (cost=160342.47..160399.69 rows=20 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)"

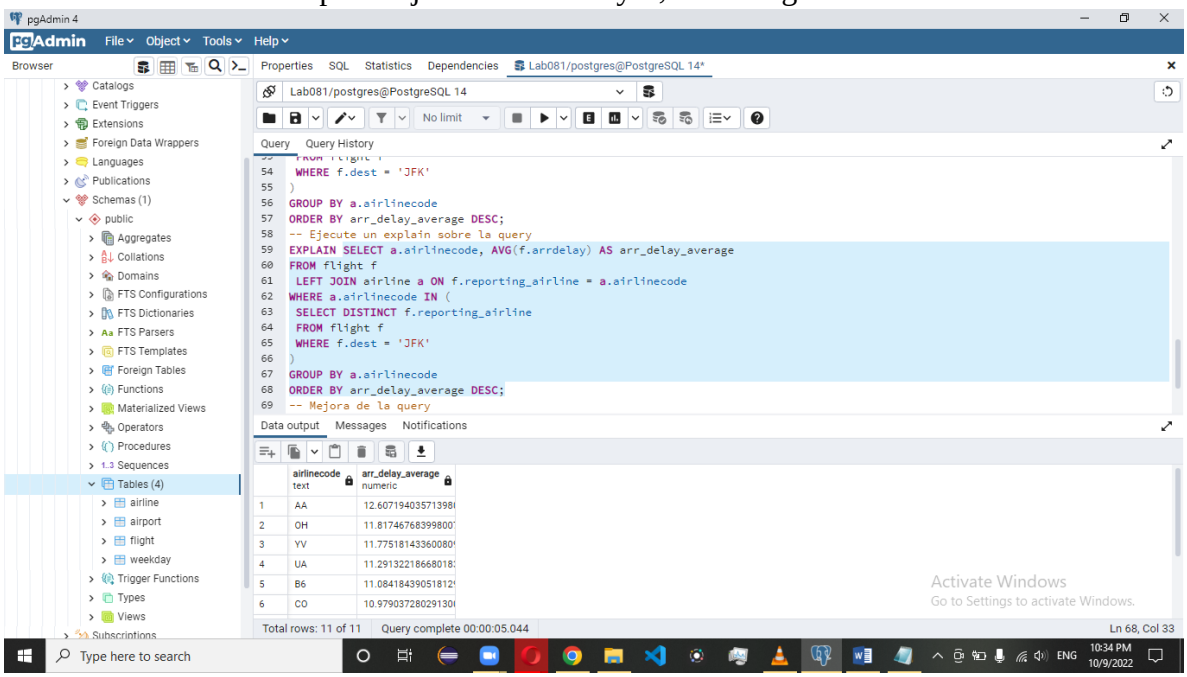
Y se tardó 4.448 segundos en ejecutarse



Crear el Nuevo index de dest y reporting airline



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



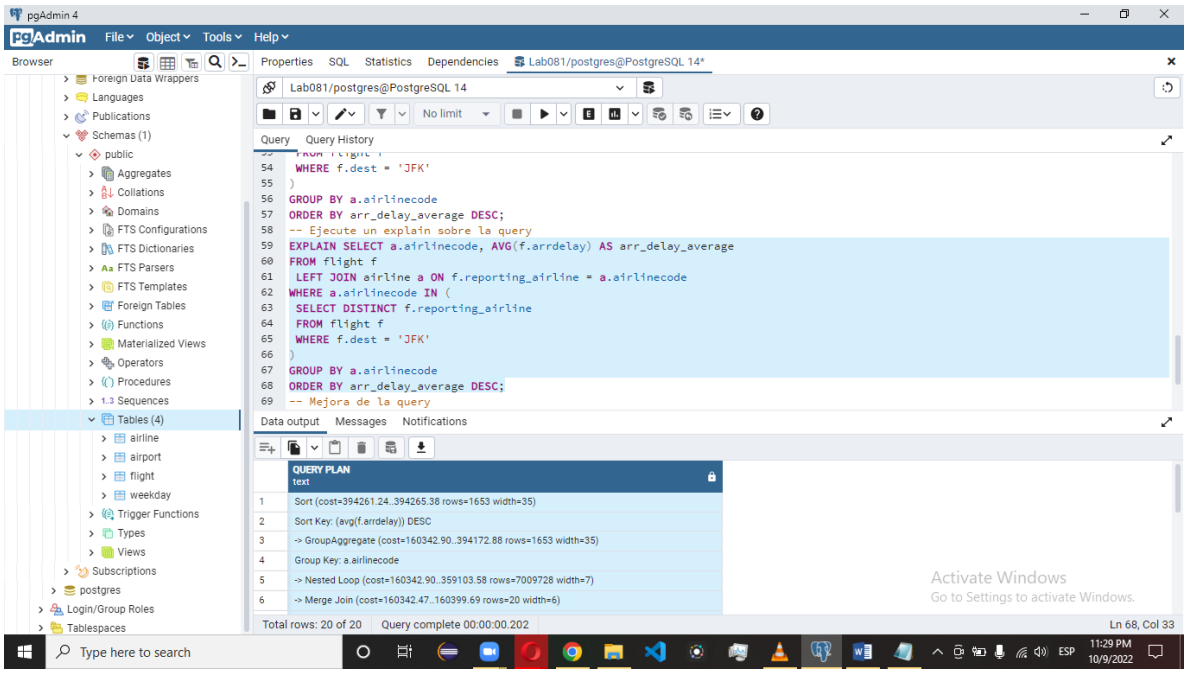
"Sort (cost=252016.17..252020.30 rows=1653 width=35)"
" Sort Key: (avg(f.arrdelay)) DESC"
" -> Finalize GroupAggregate (cost=251504.89..251927.81 rows=1653 width=35)"
" Group Key: a.airlinecode"
" -> Gather Merge (cost=251504.89..251890.61 rows=3306 width=35)"
" Workers Planned: 2"
" -> Sort (cost=250504.86..250509.00 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)"

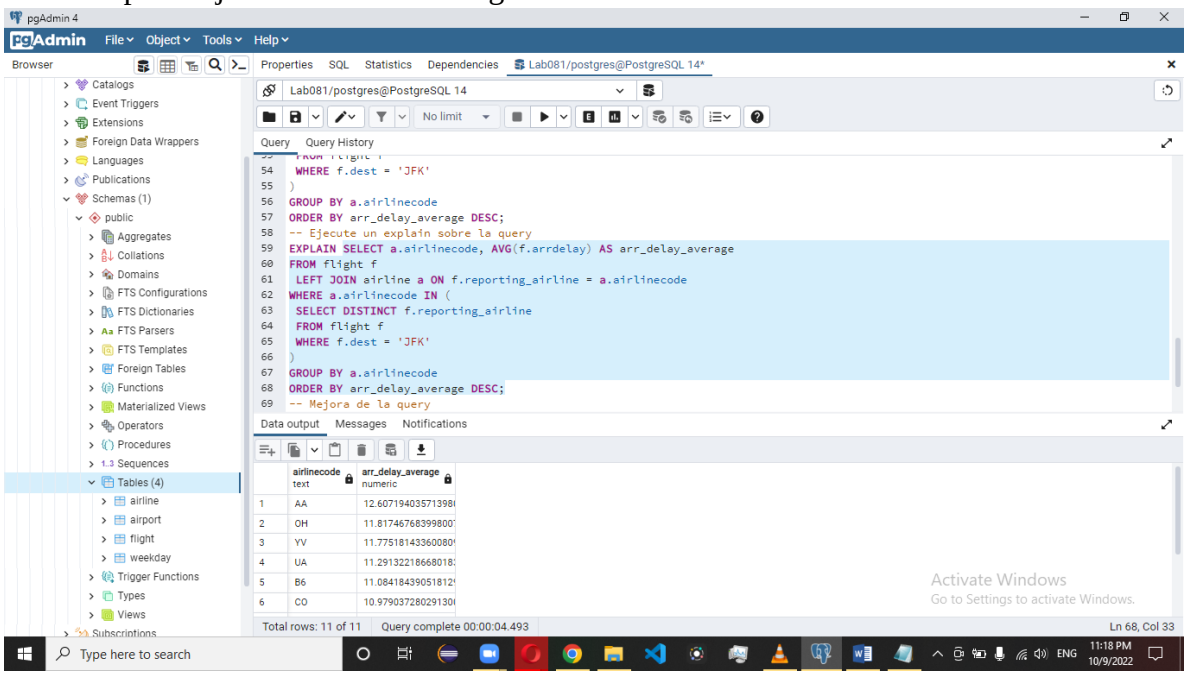
Es debido a que tiene el problema que al final termina generando un escaneo secuencial.

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)"



Y el tiempo de ejecución fue 4.493 segundos.



4. Ejercicio 4 – Consultas SARGABLES.
   a. 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
      i. Query original:
      SELECT id_curso, COUNT(*) FROM asignacion
      WHERE EXTRACT(YEAR, fecha_asignacion) = 2022
      GROUP BY id_curso

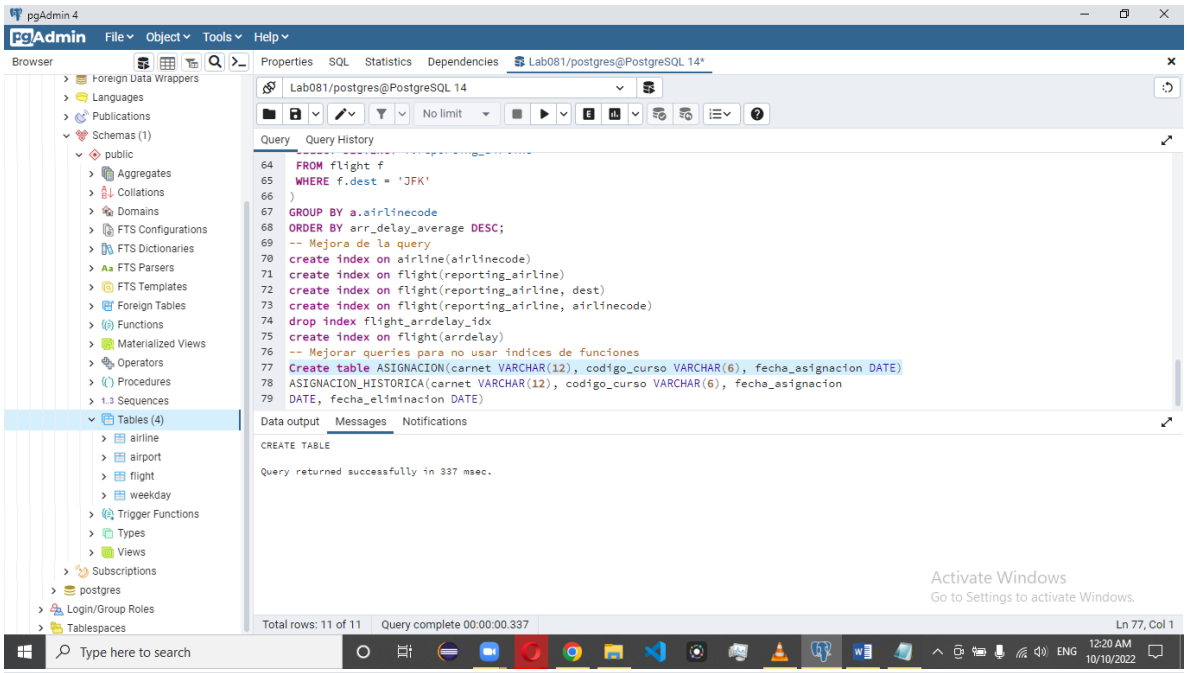
ii. SELECT id_curso, COUNT(*) FROM asignacion
WHERE fecha_asignación>’31-12-2021’ and fecha_asignacion<’01-12-2023’
GROUP BY id_curso

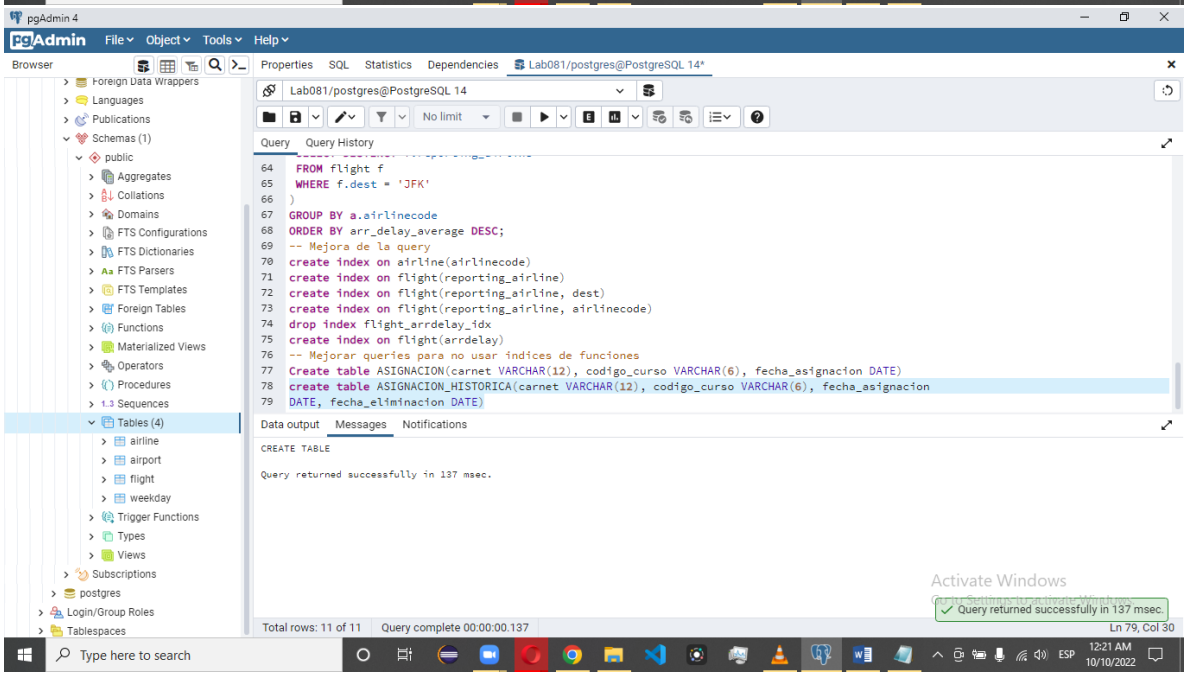
b. Esta consulta devuelve los cursos que los estudiantes con apellido “Alvarado” se asignaron, y posee un índice en los apellidos, pero no es utilizado
   i. SELECT id_curso, nombre_curso FROM asignacion
   INNER JOIN
   estudiante on asignacion.carnet = estudiante.carnet
   WHERE SUBSTRING(apellidos, 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 columa 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

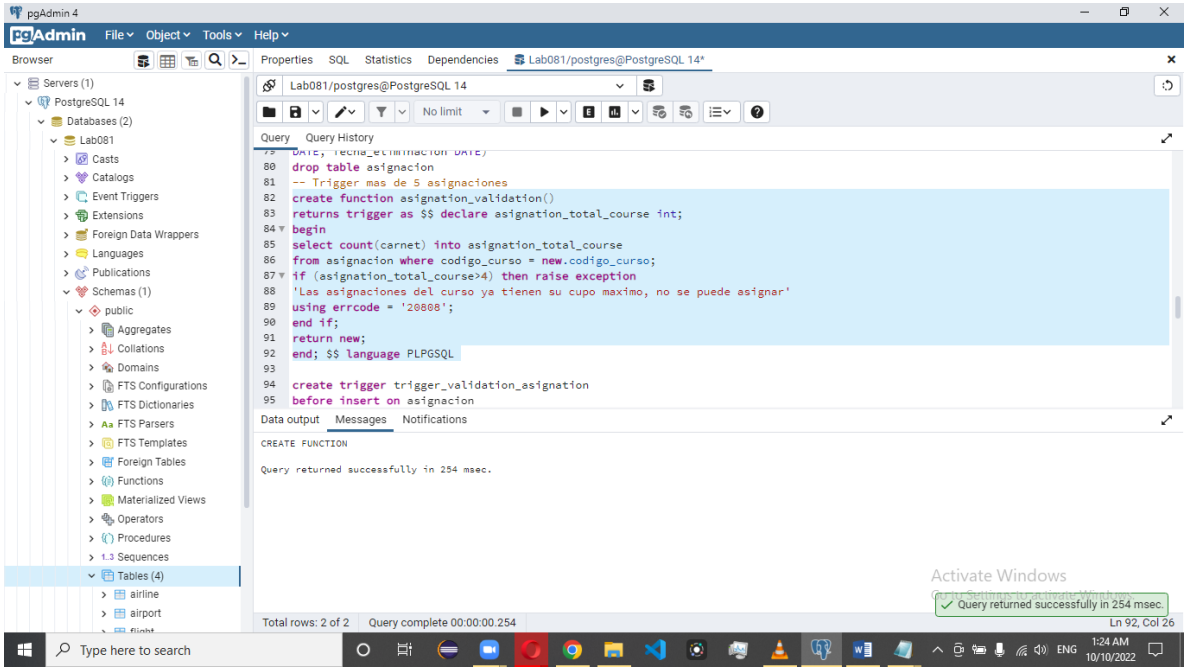
5. Ejercicio 5 – Creación de triggers.

a. Creación de tablas

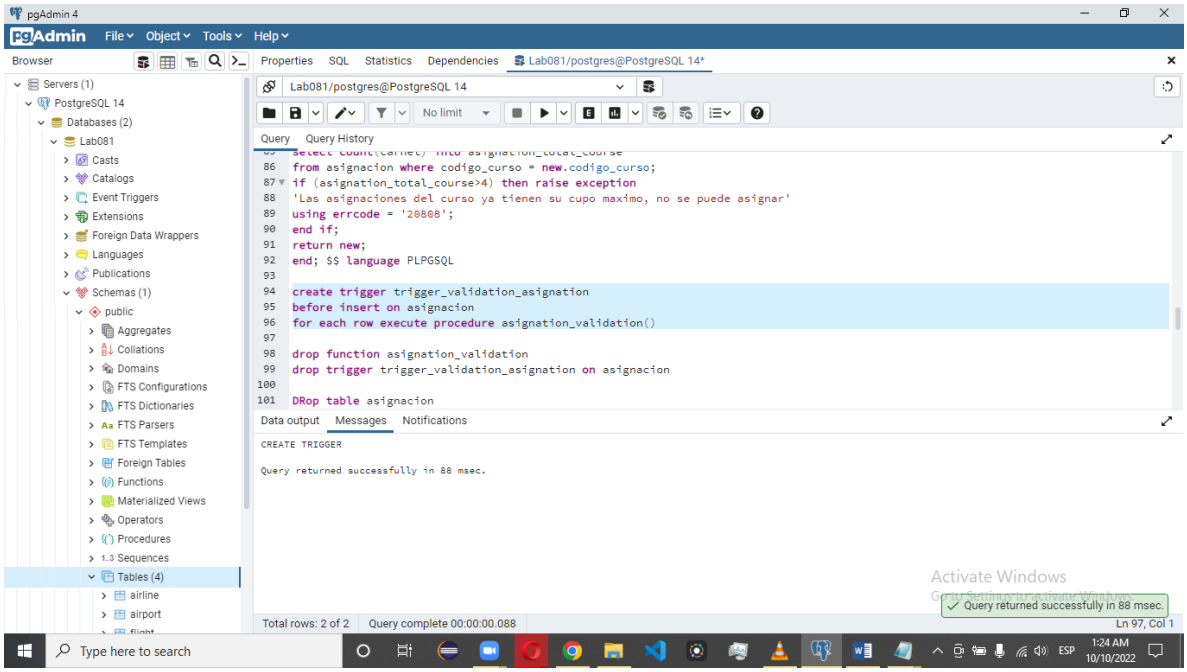




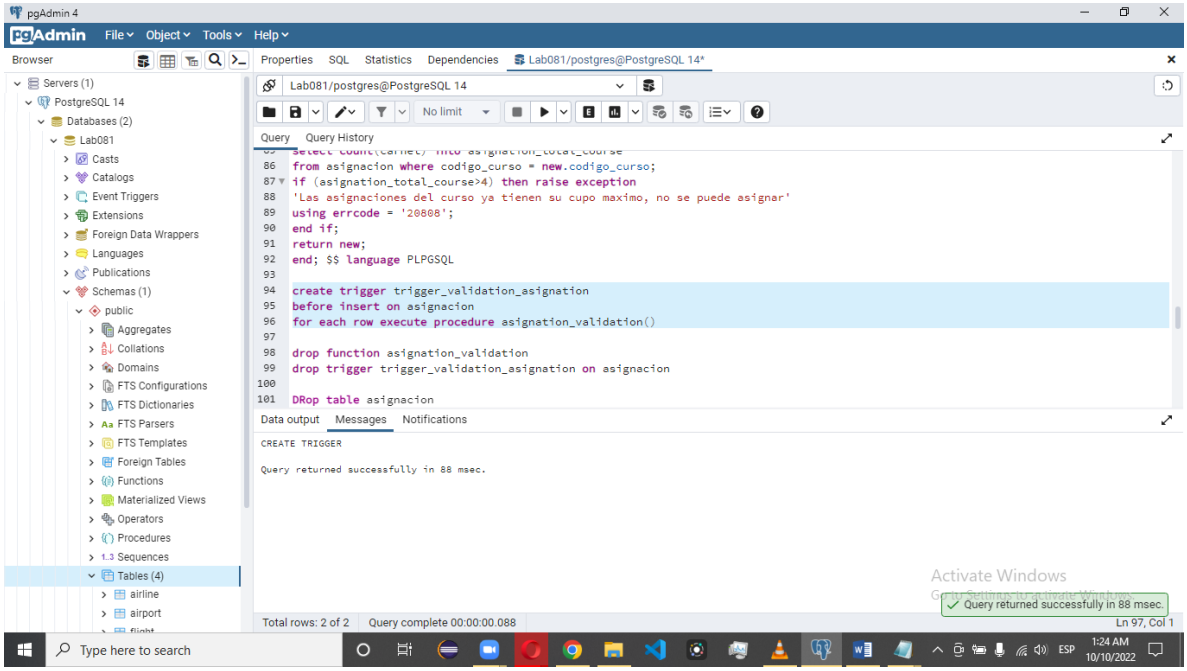
b. Crear un trigger que impida que una materia tenga más de 5 asignaciones para un mismo semestre.



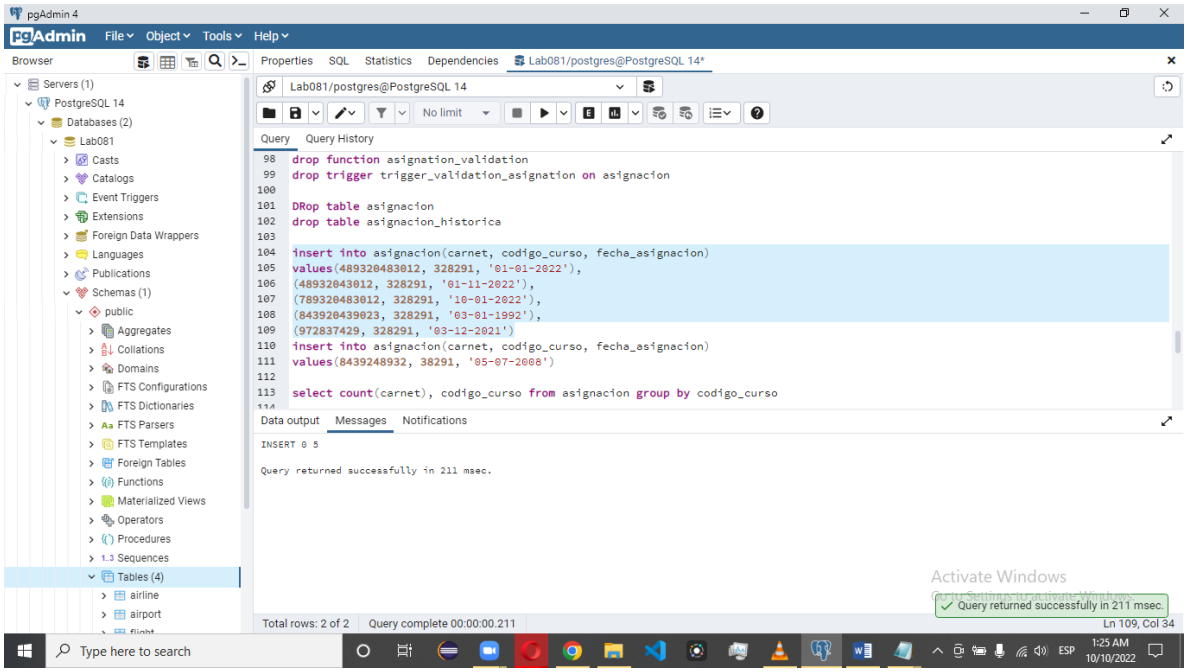
Crear la función.



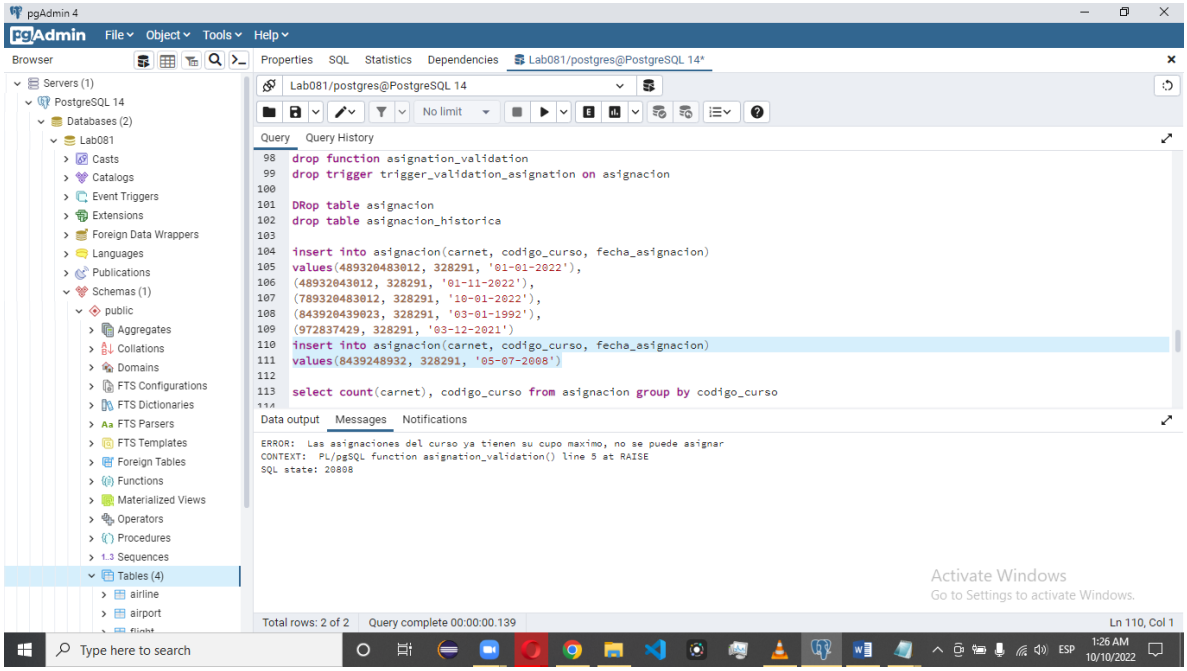
Crear el trigger.



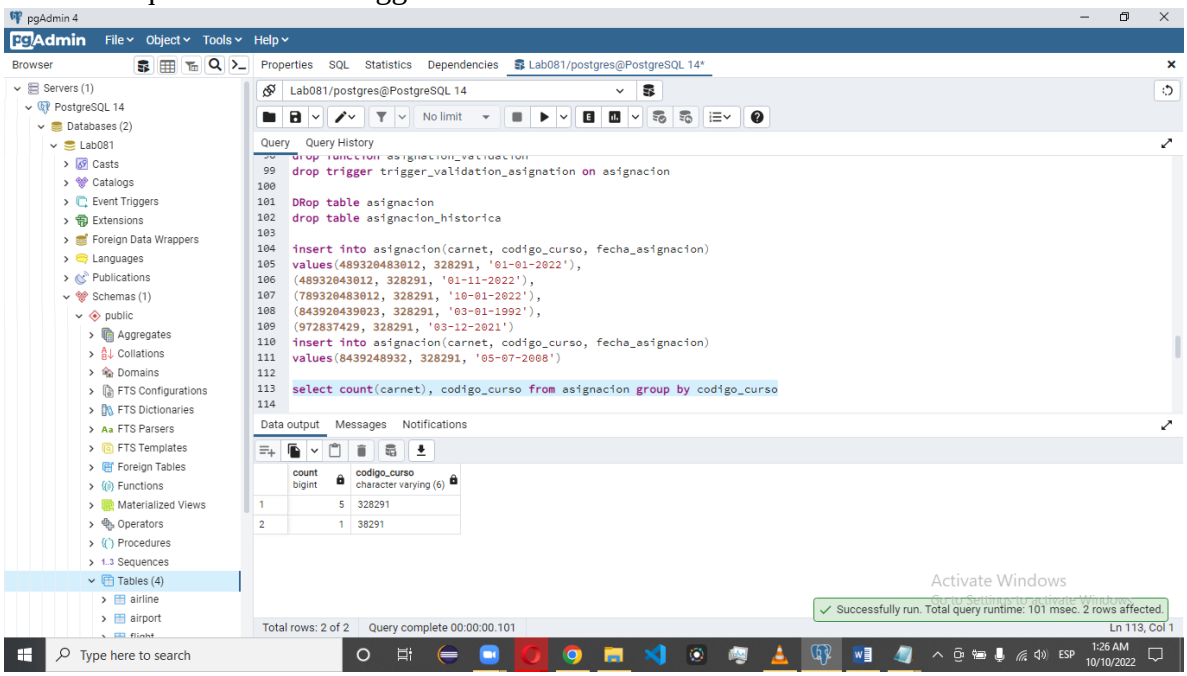
Insertar los valores



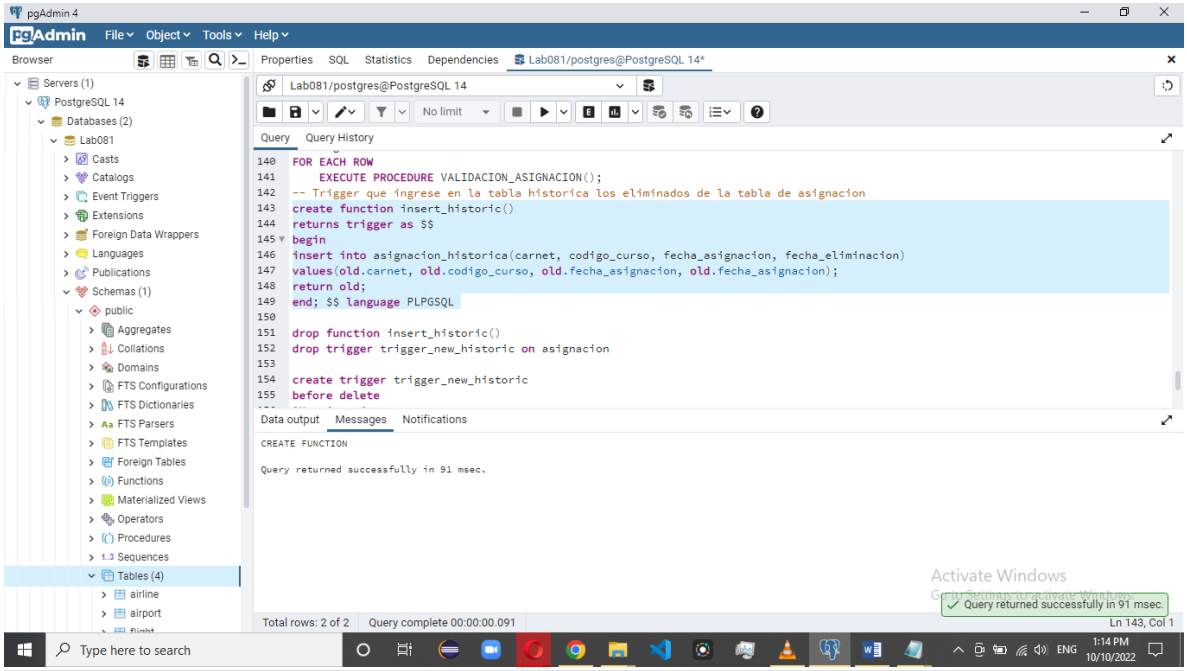
Prueba de que funcionó el trigger.

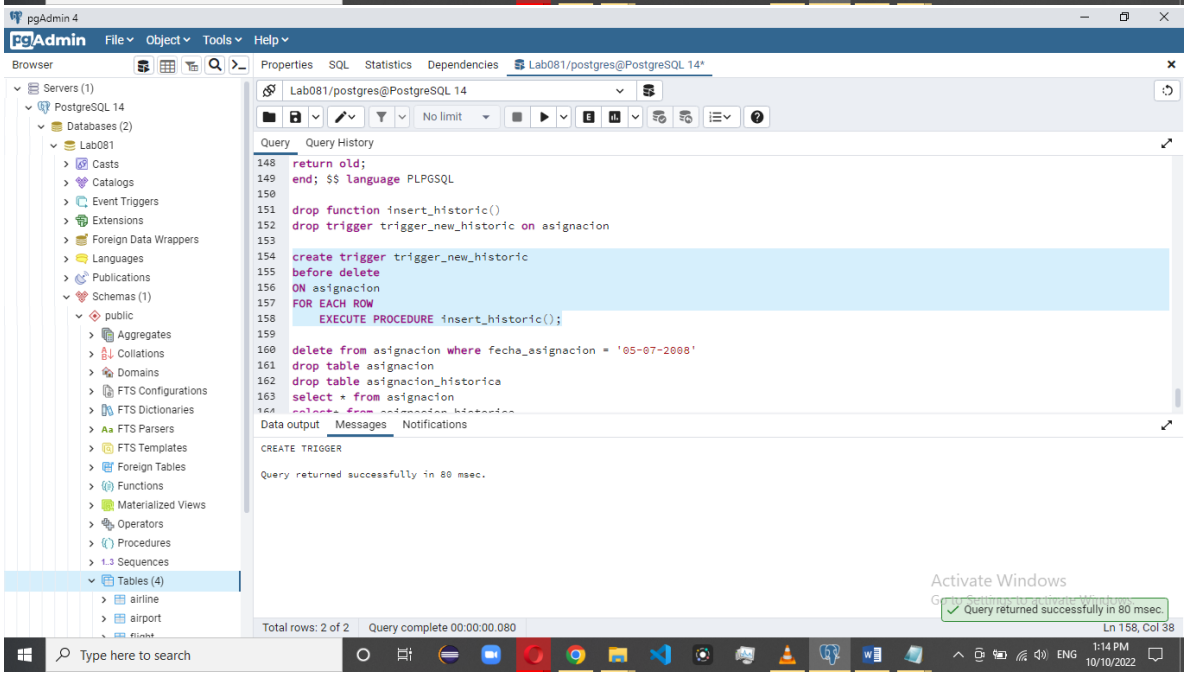


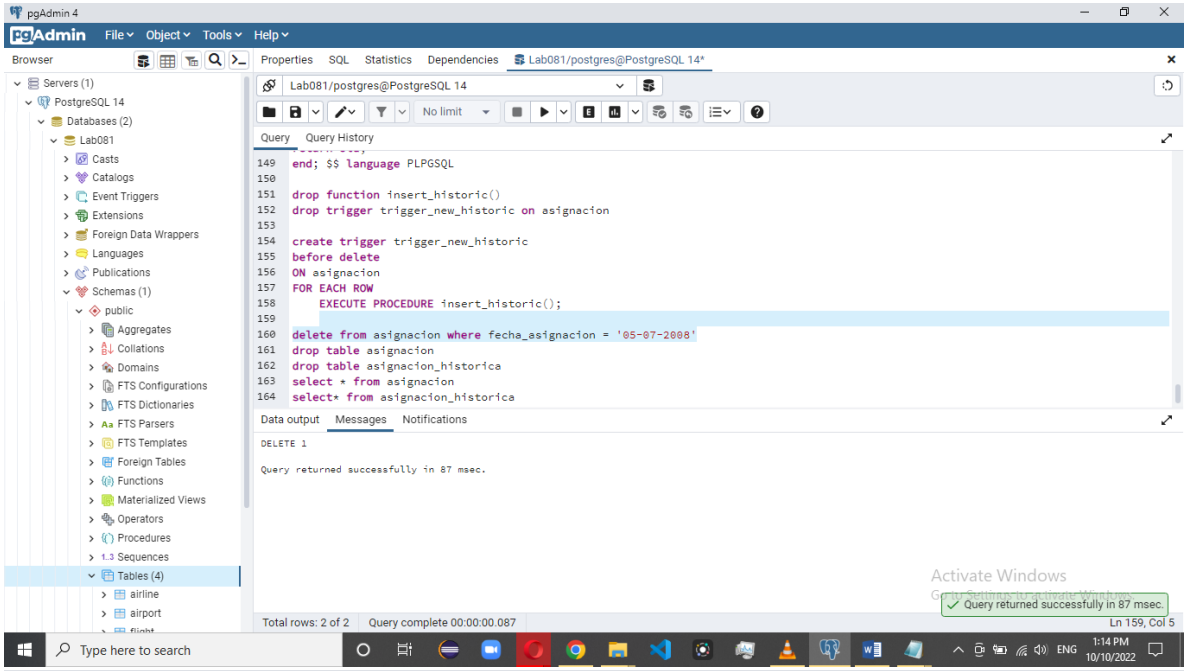
Prueba de que funciono el trigger

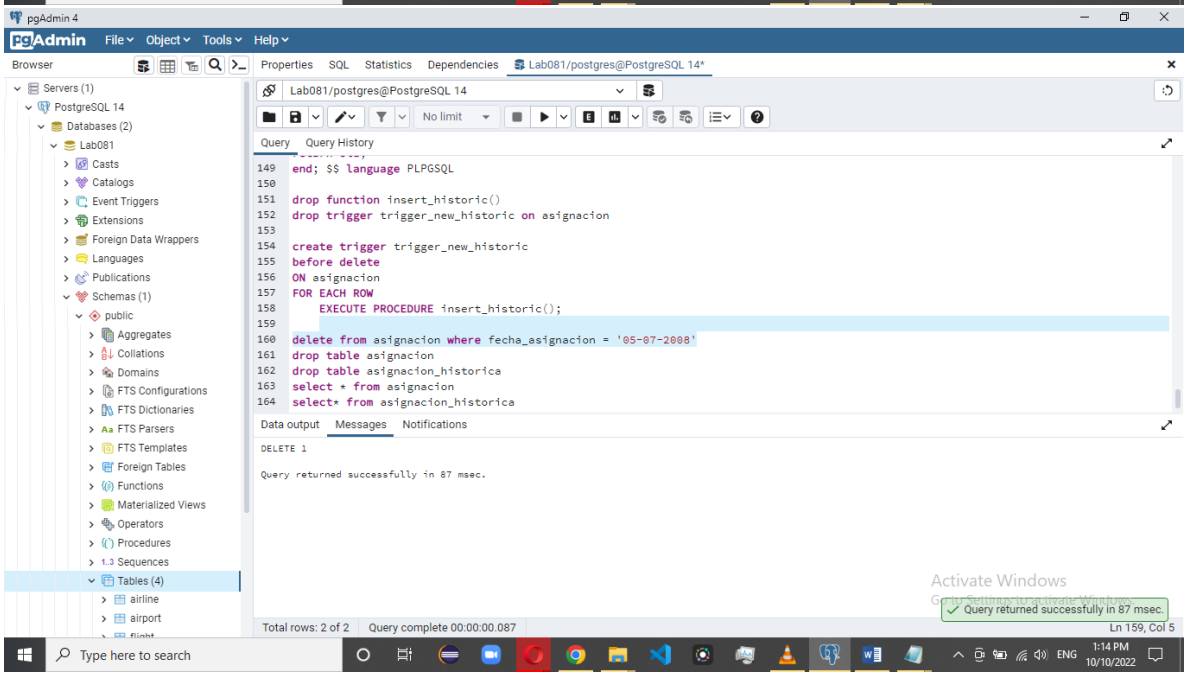


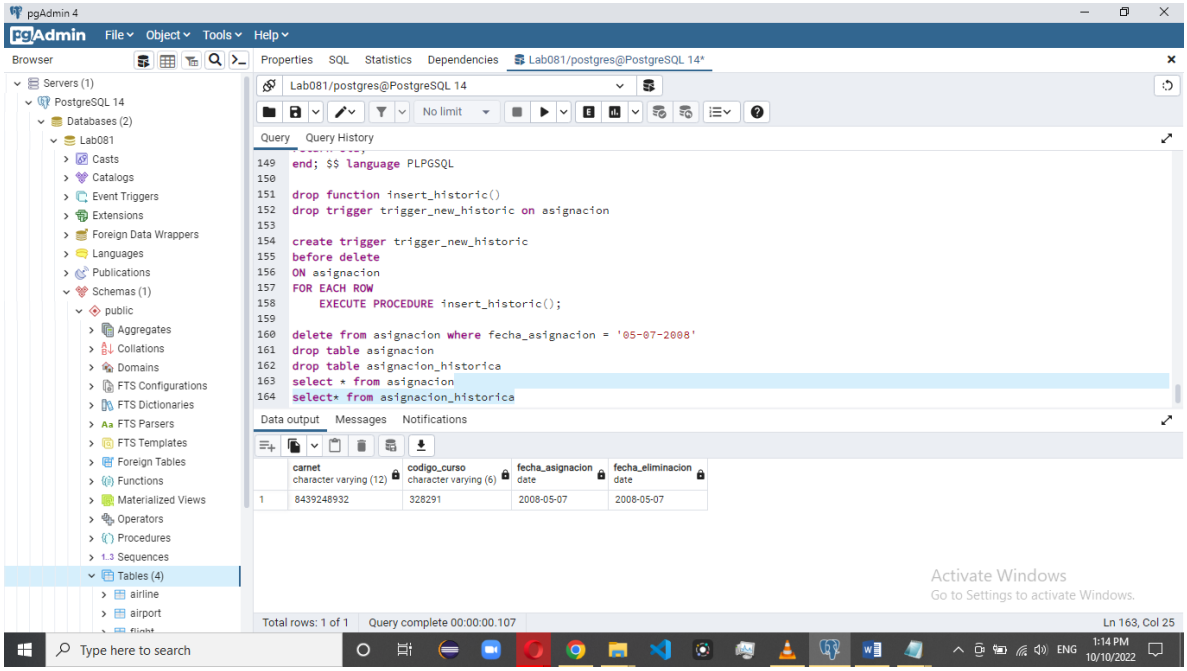
c. Crear un trigger que registre en la tabla histórica de asignaciones las asignaciones que son eliminadas.


pgAdmin 4
File Object Tools Help
Browser
Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*
Lab081/postgres@PostgreSQL 14
No limit
Query Query History
FOR EACH ROW
EXECUTE PROCEDURE VALIDACION_ASIGNACION();
-- Trigger que ingrese en la tabla historica los eliminados de la tabla de asignacion
create function insert_historic()
returns trigger as $$
begin
insert into asignacion_historica(carnet, codigo_curso, fecha_asignacion, fecha_eliminacion)
values(old.carnet, old.codigo_curso, old.fecha_asignacion, old.fecha_asignacion);
return old;
end; $$ language PLPGSQL
drop function insert_historic()
drop trigger trigger_new_historic on asignacion
create trigger trigger_new_historic
before delete
Data output Messages Notifications
CREATE FUNCTION
Query returned successfully in 91 msec.
Total rows: 2 of 2
Query complete 00:00:00.091
Query returned successfully in 91 msec.
Ln 143, Col 1
Activate Windows
Type here to search
ENG
1:14 PM
10/10/2022


pgAdmin 4
File Object Tools Help
Browser
Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*
Lab081/postgres@PostgreSQL 14
No limit
Query Query History
return old;
end; $$ language PLPGSQL
drop function insert_historic()
drop trigger trigger_new_historic on asignacion
create trigger trigger_new_historic
before delete
ON asignacion
FOR EACH ROW
EXECUTE PROCEDURE insert_historic();
delete from asignacion where fecha_asignacion = '05-07-2008'
drop table asignacion
drop table asignacion_historica
select * from asignacion
Data output Messages Notifications
CREATE TRIGGER
Query returned successfully in 80 msec.
Total rows: 2 of 2
Query complete 00:00:00.080
Query returned successfully in 80 msec.
Ln 158, Col 38
Activate Windows
Type here to search
ENG
1:14 PM
10/10/2022


pgAdmin 4
File Object Tools Help
Browser
Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*
Lab081/postgres@PostgreSQL 14
No limit
Query Query History
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
Query returned successfully in 87 msec.
Ln 159, Col 5
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
Sequences
Tables (4)
airline
airport
Type here to search
ENG
1:14 PM
10/10/2022


pgAdmin 4
File Object Tools Help
Browser
Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*
Lab081/postgres@PostgreSQL 14
No limit
Query Query History
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
Query returned successfully in 87 msec.
Ln 159, Col 5
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
Sequences
Tables (4)
airline
airport
Type here to search
ENG
1:14 PM
10/10/2022


pgAdmin 4
File Object Tools Help
Browser
Properties SQL Statistics Dependencies Lab081/postgres@PostgreSQL 14*
Lab081/postgres@PostgreSQL 14
No limit
Query Query History
end; $$ language PLPGSQL
drop function insert_historic()
drop trigger trigger_new_historic on asignacion
create trigger trigger_new_historic
before delete
ON asignacion
FOR EACH ROW
EXECUTE PROCEDURE insert_historic();
delete from asignacion where fecha_asignacion = '05-07-2008'
drop table asignacion
drop table asignacion_historica
select * from asignacion
select* from asignacion_historica
Data output Messages Notifications
carnet character varying (12)
codigo_curso character varying (6)
fecha_asignacion date
fecha_eliminacion date
1 8439248932 328291 2008-05-07 2008-05-07
Activate Windows
Go to Settings to activate Windows.
Total rows: 1 of 1
Query complete 00:00:00.107
Ln 163, Col 25
Type here to search
ENG
1:14 PM
10/10/2022