Ampliando la caja de herramientas: *common table expression* y funciones analíticas

y

El diablo está en los detalles: optimización de la base de datos en función de su uso.

NOMBRE Y APELLIDOS: **Paula Corbatón Álvarez**

EJERCICIO 1 (30%)

Apartado 1:

Código:

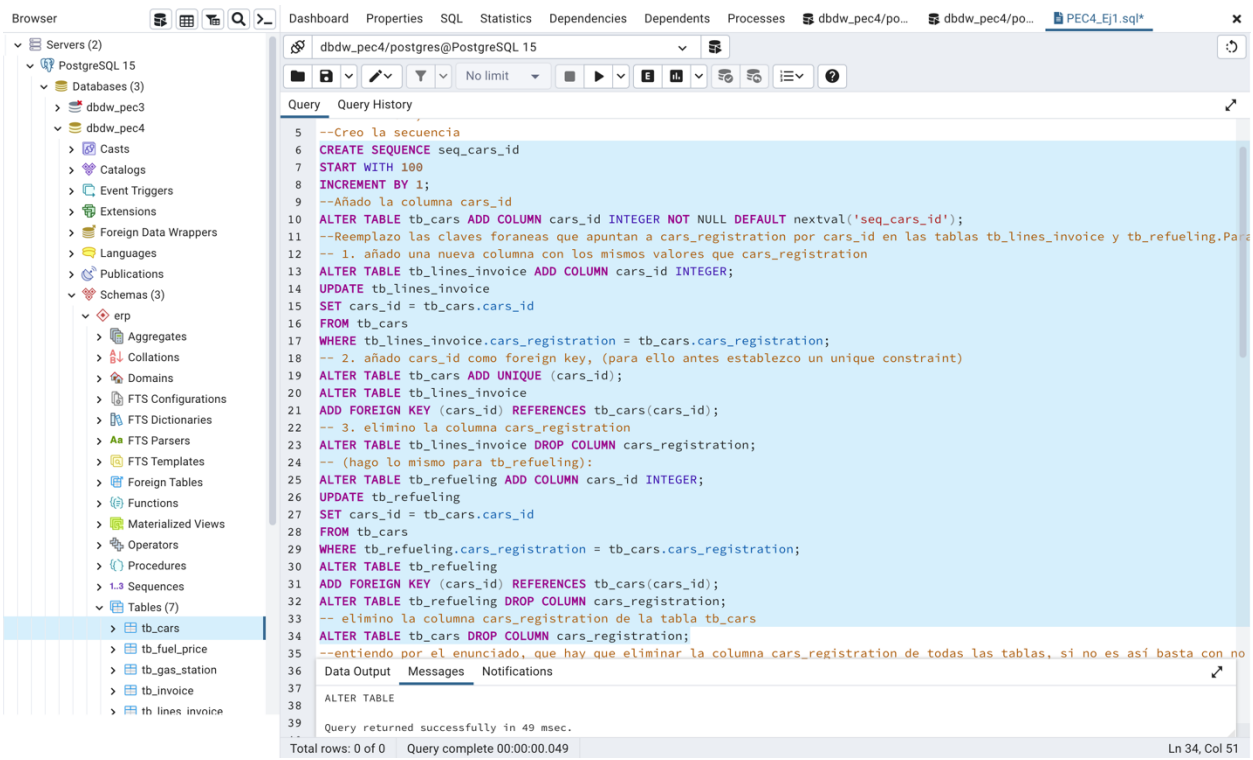


Tabla tb_cars:

```
1  SELECT * FROM erp.tb_cars;
```

Data Output | Messages | Notifications

	cars_model character varying (50)	cars_function character (20)	cars_deposit integer	cars_fuel character (10)	cars_date_input date	cars_employee character varying (50)	cars_date_registration date	cars_id integer
1	MERCEDES-VITO	reparto	60	gasoil	2022-07-22	Antonio García	2011-05-04	100
2	SKODA-FABIA	comercial	50	gasoil	2022-07-23	Manuel Rodríguez	2017-10-05	101
3	CITROEN-JUMPY	reparto	60	gasoil	2022-07-22	Juan Martínez	2010-01-08	102
4	SEAT-IBIZA	comercial	30	gasolina	2022-07-24	Ana Maria Sánchez	2019-08-12	103
5	MERCEDES-VITO	reparto	40	gasoil	2022-07-25	José Pérez	2010-09-11	104
6	BMW-X3	gerencia	60	gasolina	2022-07-26	Pablo Díaz	2016-09-08	105
7	MERCEDES-VITO	reparto	60	gasoil	2022-07-24	Rafael Muñoz	2010-04-23	106
8	MERCEDES-CITAN	reparto	60	gasoil	2022-07-24	David Ruiz	2008-06-20	107
9	SKODA-FABIA	comercial	30	gasoil	2022-07-25	Carmen Ruiz	2018-10-03	108
10	MERCEDES-VITO	reparto	60	gasoil	2022-07-26	Jesus Álvarez	2009-06-03	109
11	MERCEDES-VITO	estudio	70	gasoil	2022-07-26	Paula Corbatón	2009-06-03	110

(he añadido una nueva fila para comprobar que funcionaba)

Tabla tb_lines_invoice:

```
1  SELECT * FROM erp.tb_lines_invoice;
```

Data Output | Messages | Notifications

	inv_id [PK] integer	linv_id [PK] integer	gas_id character (5)	linv_liters integer	linv_amount numeric (12,2)	cars_id integer
1	1	1	GS01	80	159.11	106
2	1	2	GS02	96	193.88	106
3	1	3	GS03	39	73.28	106
4	1	4	GS04	44	85.32	106
5	1	5	GS05	42	89.84	106
6	1	6	GS01	95	185.09	104
7	1	7	GS02	60	120.54	104
8	1	8	GS03	28	58.80	104
9	1	9	GS04	60	119.96	104
10	1	10	GS05	65	131.03	104
11	1	11	GS01	139	279.88	102
12	1	12	GS02	144	282.36	102
13	1	13	GS03	34	64.57	102
14	1	14	GS04	36	72.68	102
15	1	15	GS05	41	87.70	102
16	1	16	GS01	125	250.60	100
17	1	17	GS02	63	123.33	100
18	1	18	GS01	104	199.95	107
19	1	19	GS02	78	154.68	107
20	1	20	GS03	42	89.84	107
21	1	21	GS04	32	66.37	107
22	1	22	GS05	35	71.23	107

Total rows: 104 of 104 | Query complete 00:00:00.152

Tabla tb_refueling:

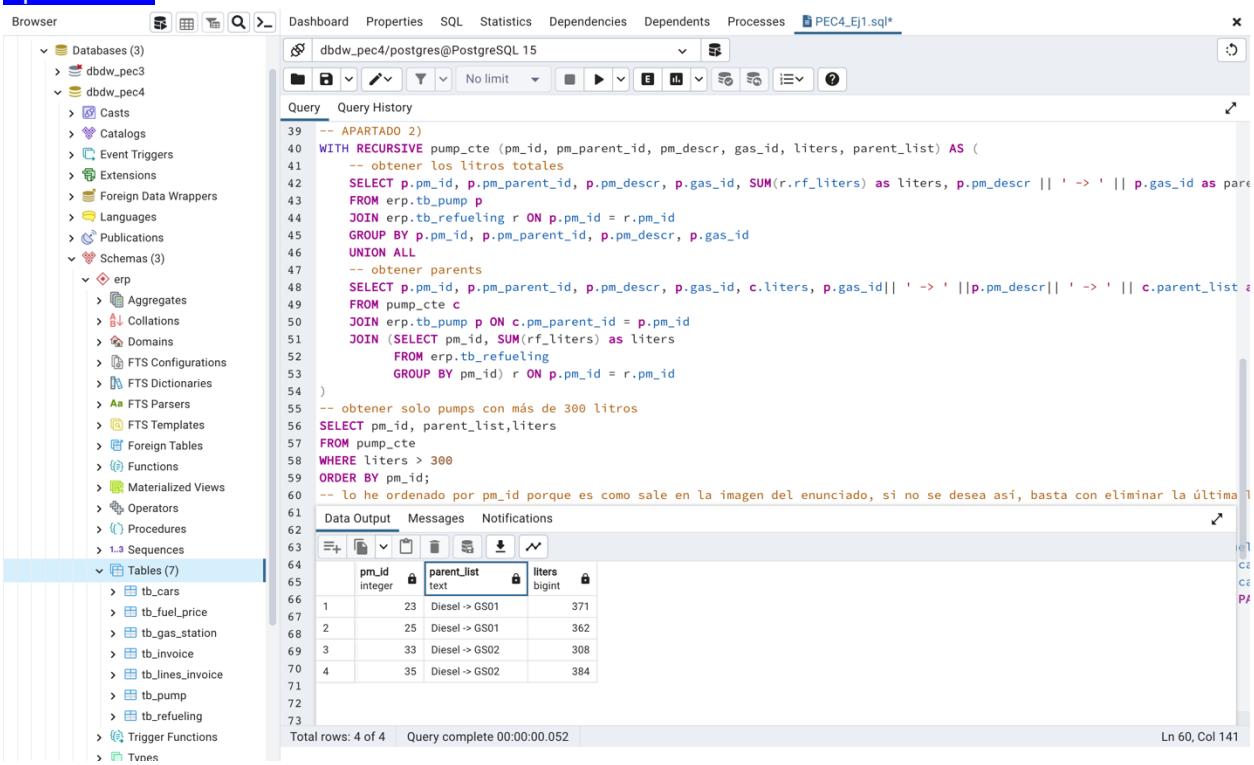
```
1  SELECT * FROM erp.tb_refueling;
```

Data Output | Messages | Notifications

	gas_id character (5)	pm_id integer	rf_liters integer	rf_date date	rf_km integer	cars_id integer
1	GS01	23	47	2022-08-01	234561	106
2	GS01	23	33	2022-08-26	235411	106
3	GS01	25	55	2022-09-05	236061	106
4	GS01	27	34	2022-09-10	236794	106
5	GS02	31	51	2022-08-06	[null]	106
6	GS02	33	45	2022-08-31	238044	106
7	GS02	33	53	2022-09-15	238794	106
8	GS03	37	39	2022-08-11	239711	106
9	GS03	41	43	2022-09-20	240277	106
10	GS04	43	44	2022-08-16	241161	106
11	GS05	51	42	2022-08-21	241877	106
12	GS01	25	32	2022-08-08	125300	104
13	GS01	29	32	2022-08-14	125816	104
14	GS01	23	31	2022-08-18	126350	104
15	GS01	25	35	2022-09-02	[null]	104
16	GS02	35	30	2022-08-11	127383	104
17	GS02	35	30	2022-08-21	127900	104
18	GS03	37	28	2022-08-24	128400	104
19	GS04	45	33	2022-08-02	128866	104
20	GS04	47	27	2022-08-27	129316	104
21	GS05	53	31	2022-08-05	129883	104
22	GS05	55	34	2022-08-30	130466	104

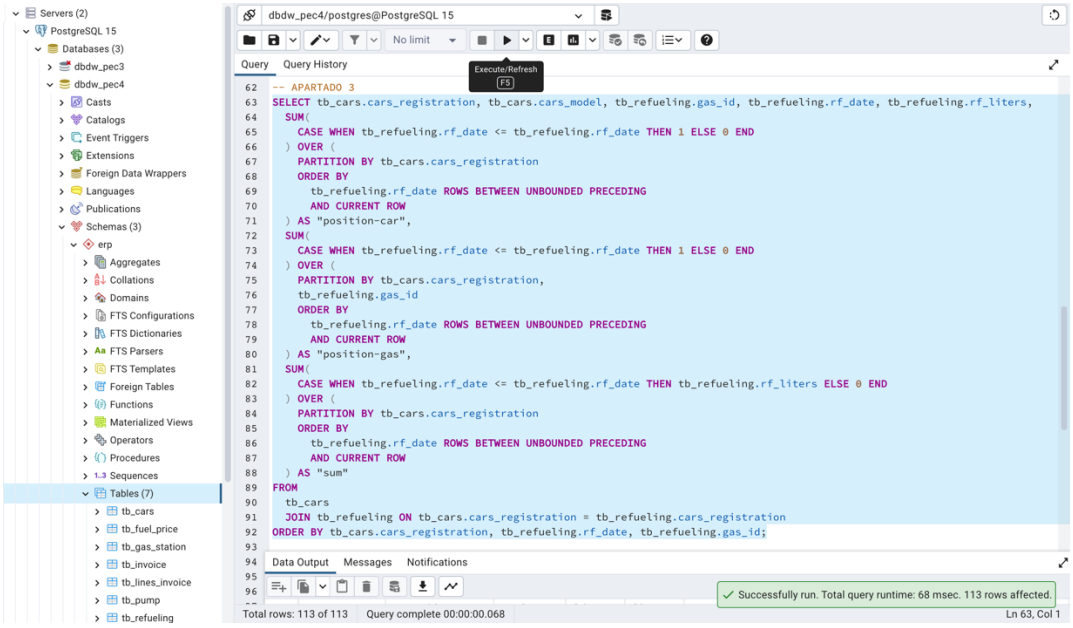
Total rows: 113 of 113 | Query complete 00:00:00.071

Apartado 2:

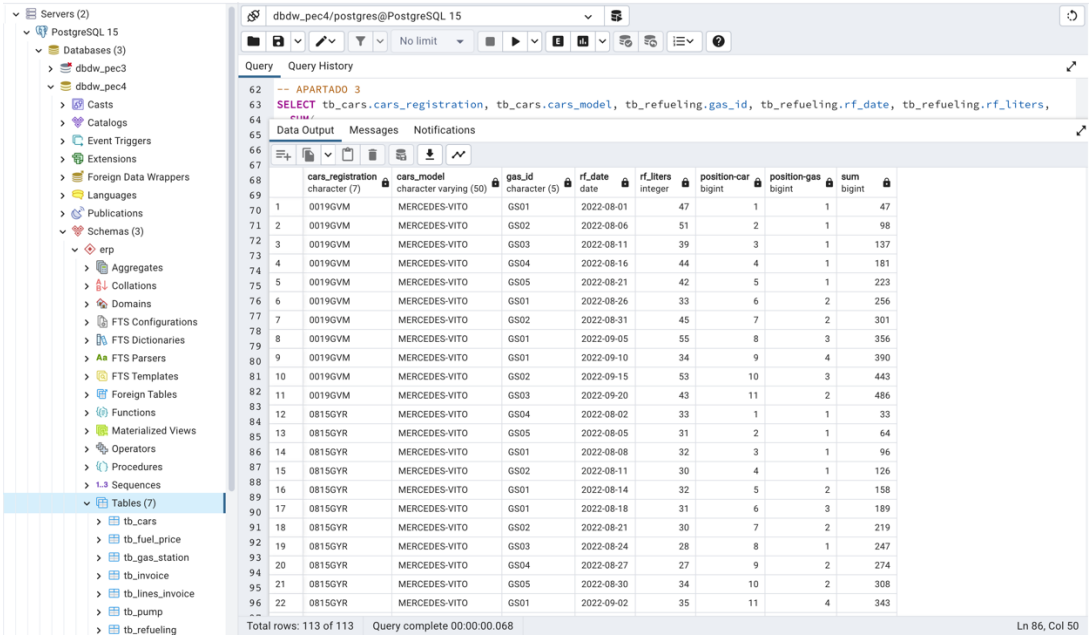

Browser
Databases (3)
dbdw_pec3
dbdw_pec4
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (3)
erp
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (7)
tb_cars
tb_fuel_price
tb_gas_station
tb_invoice
tb_lines_invoice
tb_pump
tb_refueling
Trigger Functions
Types
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
PEC4_Ej1.sql*
dbdw_pec4/postgres@PostgreSQL 15
No limit
Query
Query History
-- APARTADO 2)
WITH RECURSIVE pump_cte (pm_id, pm_parent_id, pm_descr, gas_id, liters, parent_list) AS (
-- obtener los litros totales
SELECT p.pm_id, p.pm_parent_id, p.pm_descr, p.gas_id, SUM(r.rf_liters) as liters, p.pm_descr || ' -> ' || p.gas_id as pare
FROM erp.tb_pump p
JOIN erp.tb_refueling r ON p.pm_id = r.pm_id
GROUP BY p.pm_id, p.pm_parent_id, p.pm_descr, p.gas_id
UNION ALL
-- obtener parents
SELECT p.pm_id, p.pm_parent_id, p.pm_descr, p.gas_id, c.liters, p.gas_id|| ' -> ' ||p.pm_descr|| ' -> ' || c.parent_list a
FROM pump_cte c
JOIN erp.tb_pump p ON c.pm_parent_id = p.pm_id
JOIN (SELECT pm_id, SUM(rf_liters) as liters
FROM erp.tb_refueling
GROUP BY pm_id) r ON p.pm_id = r.pm_id
)
-- obtener solo pumps con más de 300 litros
SELECT pm_id, parent_list,liters
FROM pump_cte
WHERE liters > 300
ORDER BY pm_id;
-- lo he ordenado por pm_id porque es como sale en la imagen del enunciado, si no se desea así, basta con eliminar la última
Data Output
Messages
Notifications
pm_id integer
parent_list text
liters bigint
1 23 Diesel -> GS01 371
2 25 Diesel -> GS01 362
3 33 Diesel -> GS02 308
4 35 Diesel -> GS02 384
Total rows: 4 of 4
Query complete 00:00:00.052
Ln 60, Col 141

Apartado 3:

Código:



Resultado:



EJERCICIO 2 (35%)

T1
SELECT inv_amount FROM tb_invoice WHERE inv_id=1;
COMMIT

T2
SELECT inv_amount FROM tb_invoice WHERE inv_id=1;
UPDATE tb_invoide SET inv_amout=20 WHERE inv_id=1;
ROLLBACK

a) Dirty read:
Por ejemplo:

T2:
SELECT inv_amount FROM tb_invoice WHERE inv_id=1;
>T2 obtiene el valor de inv_amount para inv_id = 1 (supongamos que es 30)
UPDATE tb_invoice SET inv_amount=20 WHERE inv_id=1;
>T2 edita el valor de in_amount = 20

-- T1 se inicia antes de que T2 ejecute ROLLBACK

T1:
SELECT inv_amount FROM tb_invoice WHERE inv_id=1;
>T1 obtiene el valor de inv_amount para inv_id = 1 (obtiene el valor editado por T2 (30))
COMMIT
>T1 guarda como valor de inv_amount = 30

T1 tomaría como valor de inv_amount el editado por T2 y esto podría dar lugar a futuros errores. Esto se puede evitar con el nivel de aislamiento "Read Commited", que se asegura de que una transacción solo pueda leer los datos que han sido **guardados** por otras transacciones

b) El hecho de que dos transacciones puedan producir interferencias no implica que se tengan que producir necesariamente. La probabilidad de interferencia depende del momento de ejecución de cada transacción y del nivel de aislamiento establecido.

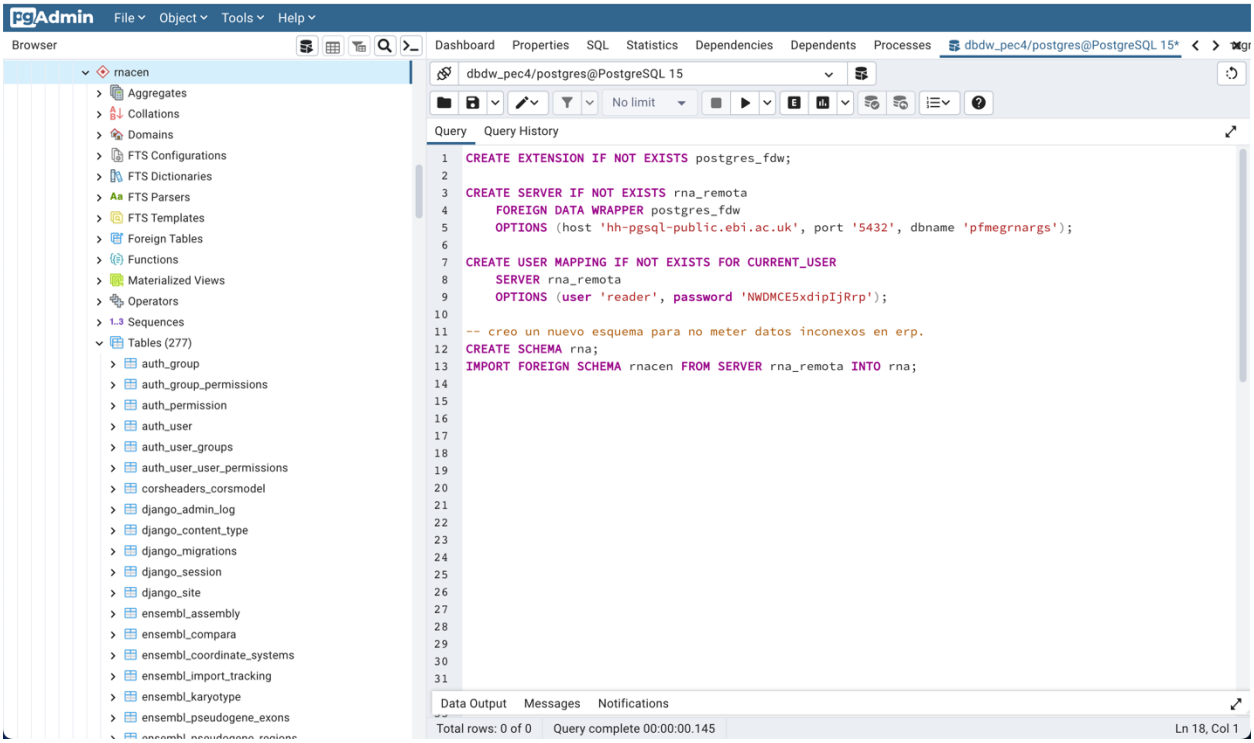
c)

Caso	Ejecución	Tipo de interferencia
1	T1 se ejecuta y completa su ejecución. Después T2 se ejecuta.	**Ningún tipo de interferencia**
2	T1 se empieza a ejecutar, T2 se empieza a ejecutar antes de que finalice T1.	Si el SELECT de T1 se ejecuta antes del UPDATE de T2: **Ningún tipo de interferencia**
		Si el SELECT de T1 se ejecuta después del UPDATE de T2 (pero antes de ROLLBACK): T1 leerá el valor actualizado de T2 resultando en un "**Dirty read**"
3	T2 se ejecuta y completa su ejecución. Después T1 se ejecuta	**Ningún tipo de interferencia**

EJERCICIO 3 (20%)

Foreign Data Wrapper for PostgreSQL (postgres_fdw) es una extensión de PostgreSQL que permite acceder a los datos almacenados en otros servidores PostgreSQL como si estuvieran en una tabla local. Esto nos permite crear tablas foráneas en nuestro servidor PostgreSQL local (estas tablas están vinculadas a otra tabla que existe en otro servidor PostgreSQL diferente). Esto es muy útil puesto que nos permite consultar y manipular datos que están almacenados en el servidor remoto como si estuvieran almacenados localmente, también nos permite enviar ciertas operaciones al servidor externo y así mejorar el rendimiento.

Ejemplo mínimo:
Una vez tenemos la extensión instalada en ambos servidores:



EJERCICIO 4 (15%)

```
1  SET search_path TO erp;
2  EXPLAIN SELECT * FROM tb_refueling r INNER JOIN tb_cars c
3  ON c.cars_registration = r.cars_registration
```

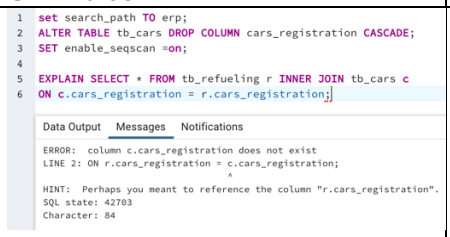
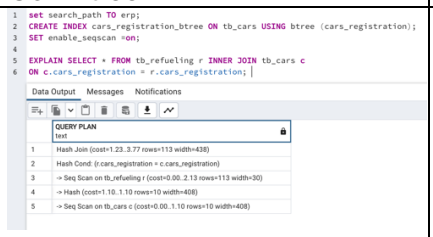
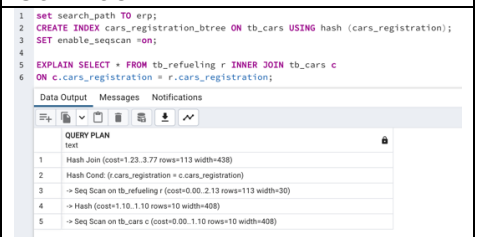
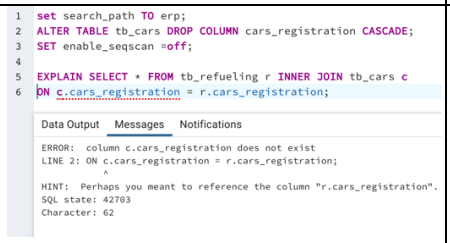
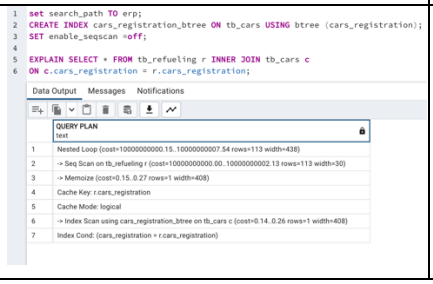
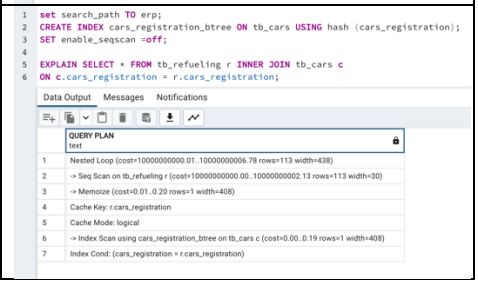
Data Output | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.15..11.30 rows=113 width=438)
2	-> Seq Scan on tb_refueling r (cost=0.00..2.13 rows=113 width=30)
3	-> Memoize (cost=0.15..0.60 rows=1 width=408)
4	Cache Key: r.cars_registration
5	Cache Mode: logical
6	-> Index Scan using pk_cars on tb_cars c (cost=0.14..0.59 rows=1 width=408)
7	Index Cond: (cars_registration = r.cars_registration)

Voy a describir el plan por líneas:

Fila/s:	Descripción:
1	Es un bucle anidado, por lo tanto la base de datos iterará sobre las filas de tb_refueling y para cada fila buscará la fila correspondiente de tb_cars (esto lo hará sabiendo que ambas tablas se unen por cars_registration)
2	"Seq Scan" en la tabla tb_refueling significa que la base de datos está escaneando secuencialmente todas las filas de la tabla. (que lo haga secuencialmente significa que lo está haciendo en el orden en el que las filas están almacenadas físicamente en el disco)
3, 4, 5	Muestran los detalles del escaneo de la tabla tb_cars. "Memoize" significa que la base de datos está utilizando el caché para buscar las filas coincidentes (esto significa que reutiliza los resultados de escaneos anteriores), Cache Key: r.cars_registration significa que la base de datos utiliza la columna cars_registration de la tabla tb_refueling como clave para identificar las filas en el cache. Cache Mode: logical, significa que el caché no almacena datos reales, solo el resultado de la consulta.
6, 7	Muestran los detalles del "index scan" en la tabla tb_cars. Esto significa que la base de datos utiliza el índice denominado "pk_cars" en la columna cars_registration para buscar las filas coincidentes en la segunda tabla. "index cond" es la condición utilizada por el índice "pk_cars" para encontrar las filas (cars_registration= cars_registration)

b)

	Sin índice	Con Btree	Con Hash
seqscan =on			
seqscan =off			

Criterios de valoración

En el enunciado se indica el peso/valoración de cada ejercicio.
Para conseguir la puntuación máxima en los ejercicios, es necesario explicar con claridad la solución que se propone.

Formato y fecha de entrega

Tenéis que enviar la PEC al buzón de Entrega y registro de EC disponible en el aula (apartado Evaluación). El formato del archivo que contiene vuestra solución puede ser .pdf, .doc y .docx. Para otras opciones, por favor, contactar previamente con vuestro consultor. El nombre del fichero debe contener el código de la asignatura, vuestro apellido y vuestro nombre, así como el número de actividad (PEC4).
La fecha límite para entregar la PEC es el **23/01/2023**.

Nota: **Propiedad intelectual**

Al presentar una práctica o PEC que haga uso de recursos ajenos, se tiene que presentar junto con ella un documento en que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y su estatus legal: si la obra está protegida por el copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL etc.). El estudiante tendrá que asegurarse que la licencia que sea no impide específicamente su uso en el marco de la práctica o PEC. En caso de no encontrar la información correspondiente tendrá que asumir que la obra está protegida por el copyright.

Será necesario, además, adjuntar los ficheros originales cuando las obras utilizadas sean digitales, y su código fuente, si así corresponde.