Procedimientos almacenados y disparadores, ¿para qué son necesarios?

NOMBRE Y APELLIDOS: Jorge Oliva Ramos

La empresa "UOC Salud", ahora que dispone de la base de datos que hemos construido en la UOC como parte de la asignatura **Bases de datos para *Data Warehousing***, ha obtenido el presupuesto necesario para implementar una serie de mejoras. De nuevo, se ha puesto en contacto con nosotros para que implementéis los requisitos que nos han propuesto.

Para la implementación de esta PEC, debéis de crear una base de datos nueva denominada **BD_DW3** y ejecutar el script adjunto **BBDD_Clinical_structures.sql**. El segundo script proporcionado, **BBDD_Clinical_data.sql,** os dará un conjunto de datos que os permitirá implementar los componentes requeridos en los diferentes apartados de esta PEC.

Consideraciones para la entrega y realización de la PEC:

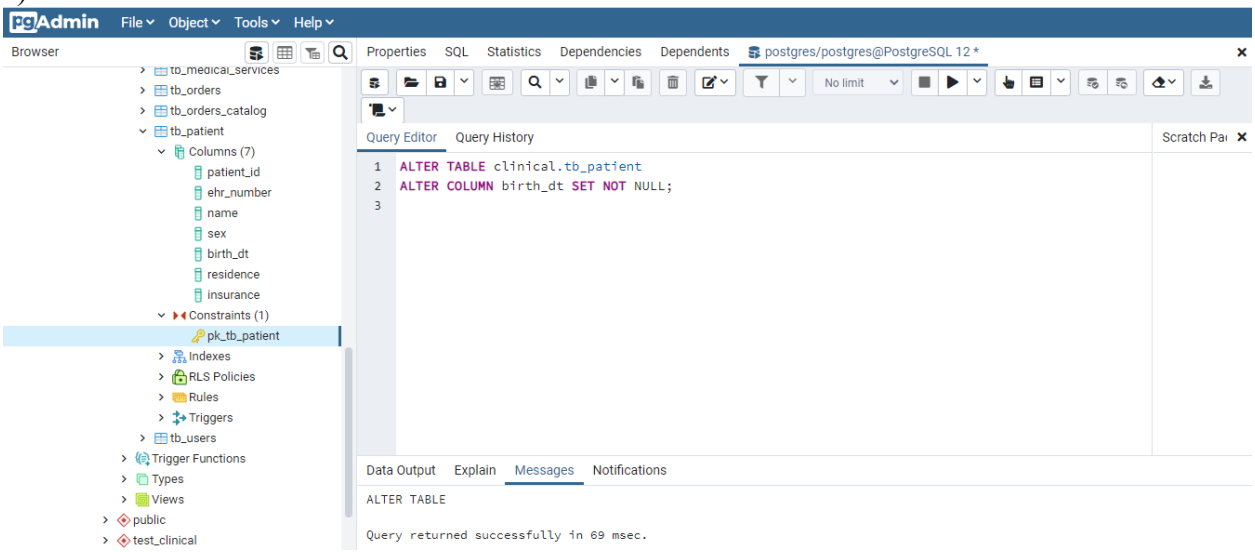
- Todo lo que se pide en esta PEC está explicado en los bloques didácticos 2 y 3 (salvo que se trate de un ejercicio de investigación, cuyo enunciado lo especificará). No es necesario adelantar el estudio del material de otros bloques didácticos para la realización de esta PEC.
- En esta PEC trabajaremos procedimientos/funciones y disparadores. Al tratarse de objetos más complejos, debéis de asegurar una correcta ejecución de estos. Se recomienda que probéis de forma exhaustiva los componentes programados y os creéis vuestros casos de prueba utilizando la base de datos proporcionada.
- Se recomienda la utilización de **pgAdmin** para la implementación de toda la PEC. Existe otra alternativa que es ***psql*** (línea de comandos), pero es preferible que utilicéis pgAdmin ya que es una interfaz gráfica que os permitirá editar y crear sentencias SQL (así como mostrar los resultados) de forma más sencilla que *psql*.
- Tal y como se indica en el enunciado, cada respuesta a los ejercicios ha de entregarse en un fichero *.sql* diferente, con el nombre correspondiente. Se evaluará el código entregado en estos ficheros *.sql* y **NO el código que aparezca en el documento o en los pantallazos adjuntos**.
- Las capturas de pantalla de los ejercicios (y explicaciones pertinentes) han de proporcionarse en un documento aparte (se proporciona una plantilla para el caso, **indicad vuestro nombre en el documento**, por favor).
- Se debe de realizar la entrega de todos los ficheros de la PEC (tanto los ficheros *.sql* como el documento con explicaciones y capturas de pantalla) en un fichero comprimido *.zip*.

Consideraciones para la evaluación del ejercicio:

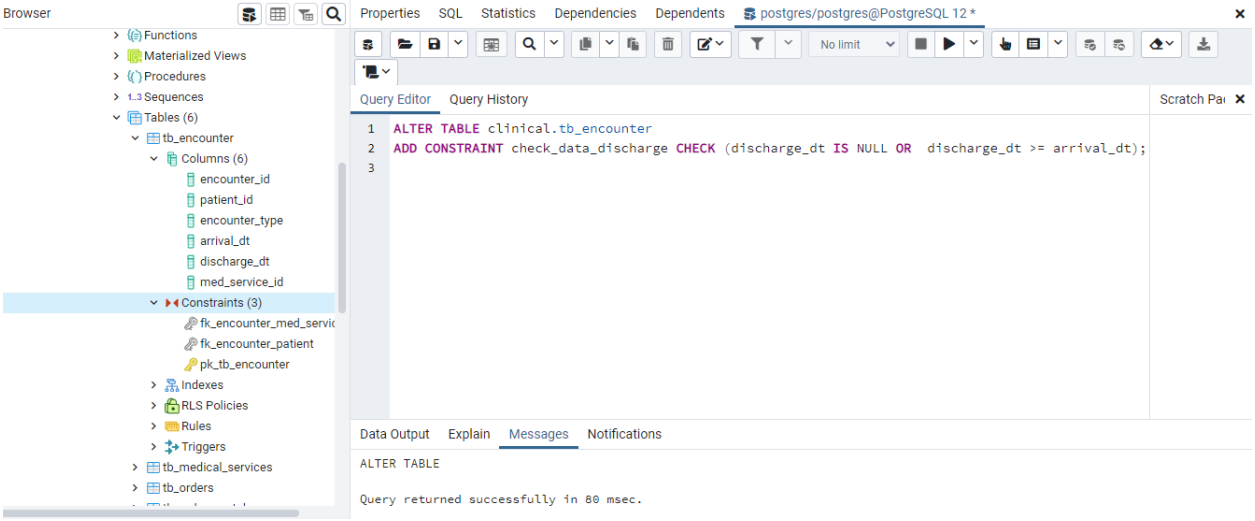
- Se tendrá en cuenta la aplicación de las buenas prácticas de codificación en SQL, de consultas y de programación de procedimientos y disparadores. Es decir: código con sangrado, uso de cláusulas SQL de forma correcta, comentarios, cabeceras en el procedimiento, etc.
- Los *scripts* proporcionados por el estudiante con las soluciones de los ejercicios han de ejecutarse correctamente. El estudiante ha de asegurarse de que lanzando el *script* completo de cada ejercicio no produzca ningún error.
- **Importante**: Las sentencias SQL proporcionadas en los *scripts* han de ser creadas de forma manual y no mediante asistentes que PostgreSQL/pgAdmin puedan proporcionar. Se pretende aprender SQL y no la utilización de asistentes.
- Las sentencias SQL proporcionadas en los ejercicios han de ser **solamente** aquellas que pide el enunciado y ninguna otra más. Cualquier sentencia añadida a mayores, si está mal o provoca que el *script* no se ejecute correctamente a la hora de corregirlo, penalizará el ejercicio.

EJERCICIO 1 (20%)

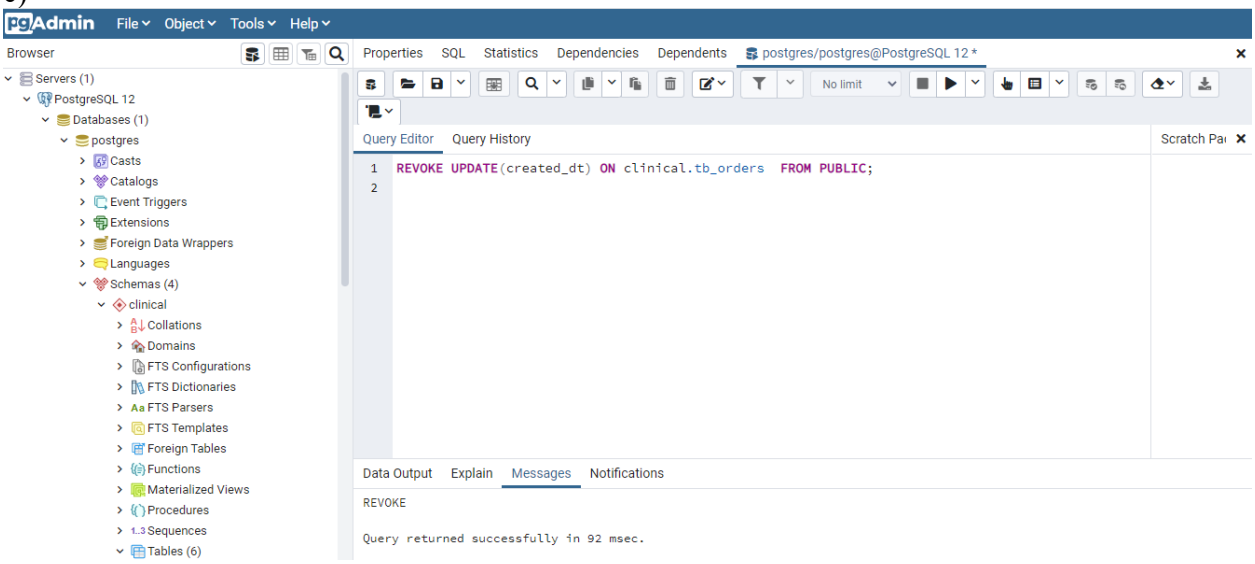
a)

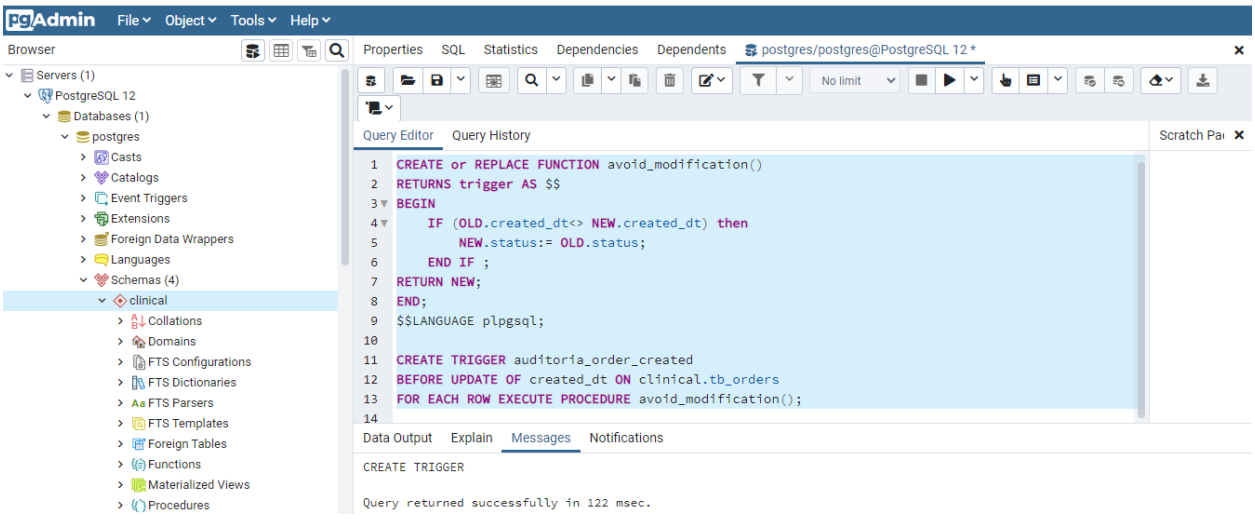


b)



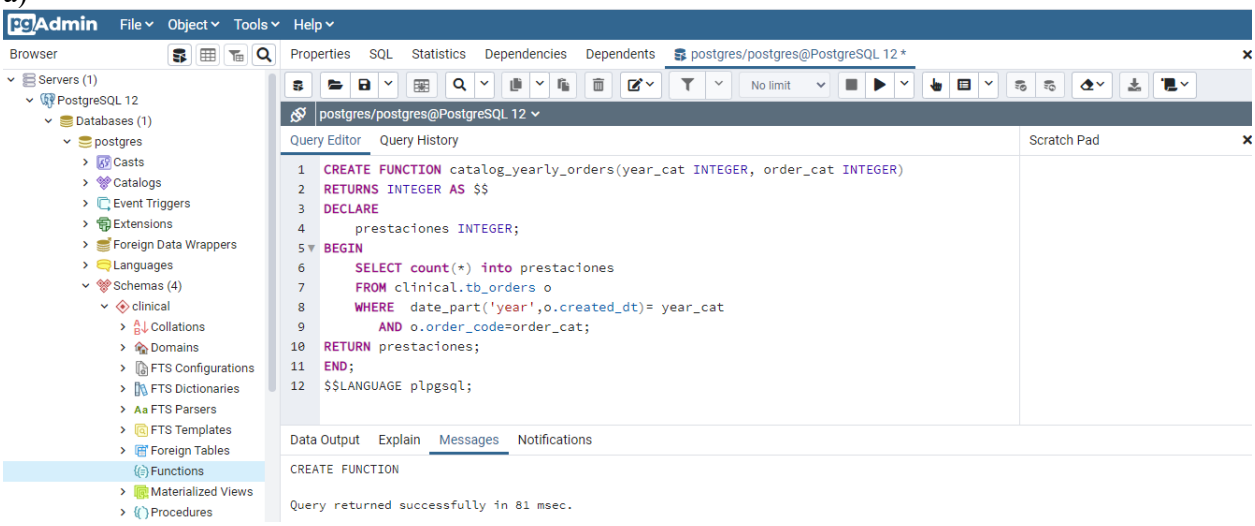
c)


pgAdmin
File
Object
Tools
Help
Browser
Properties
SQL
Statistics
Dependencies
Dependents
postgres/postgres@PostgreSQL 12 *
No limit
Query Editor
Query History
Scratch Pa
1 REVOKE UPDATE(created_dt) ON clinical.tb_orders FROM PUBLIC;
2
Data Output
Explain
Messages
Notifications
REVOKE
Query returned successfully in 92 msec.
Servers (1)
PostgreSQL 12
Databases (1)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas (4)
clinical
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (6)

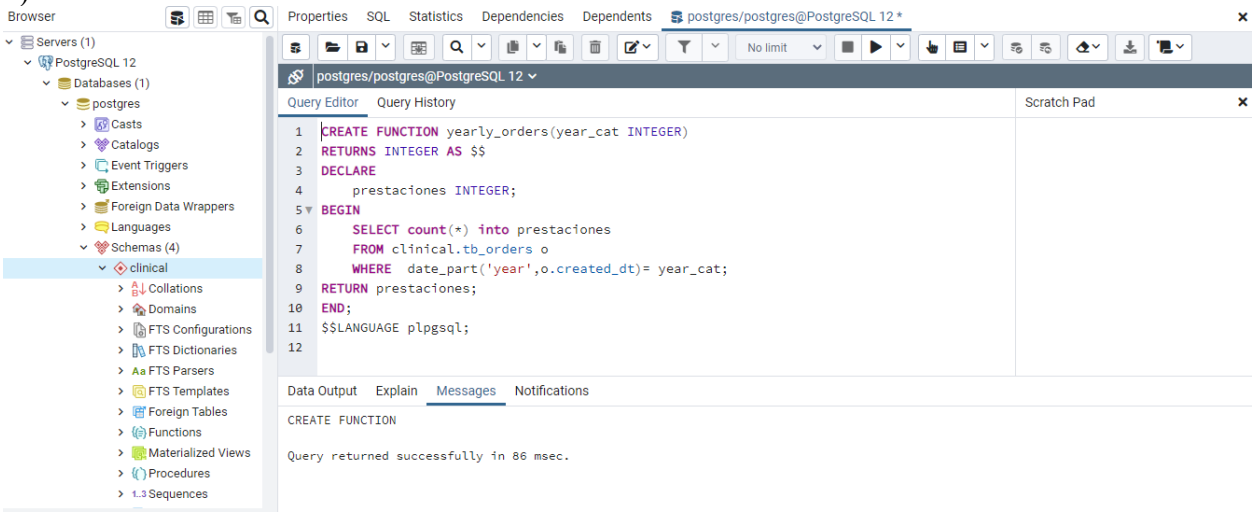

pgAdmin
File
Object
Tools
Help
Browser
Properties
SQL
Statistics
Dependencies
Dependents
postgres/postgres@PostgreSQL 12 *
No limit
Query Editor
Query History
Scratch Pa
1 CREATE or REPLACE FUNCTION avoid_modification()
2 RETURNS trigger AS $$
3 BEGIN
4 IF (OLD.created_dt<> NEW.created_dt) then
5 NEW.status:= OLD.status;
6 END IF ;
7 RETURN NEW;
8 END;
9 $$LANGUAGE plpgsql;
10
11 CREATE TRIGGER auditoria_order_created
12 BEFORE UPDATE OF created_dt ON clinical.tb_orders
13 FOR EACH ROW EXECUTE PROCEDURE avoid_modification();
14
Data Output
Explain
Messages
Notifications
CREATE TRIGGER
Query returned successfully in 122 msec.
Servers (1)
PostgreSQL 12
Databases (1)
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas (4)
clinical
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures

EJERCICIO 2 (40%)

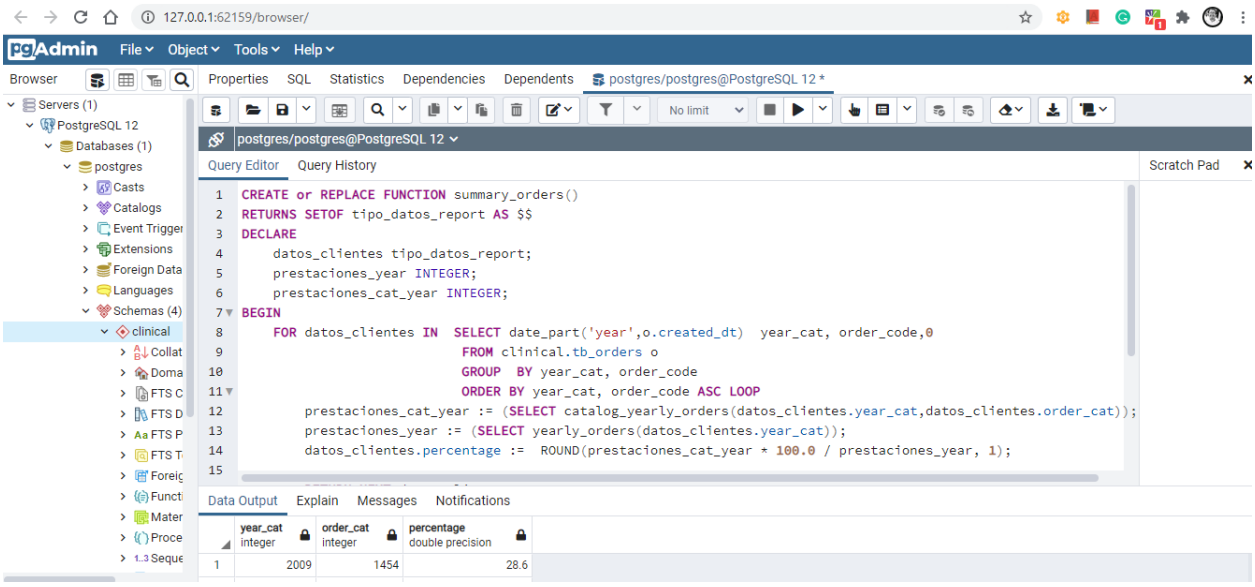
a)



b)

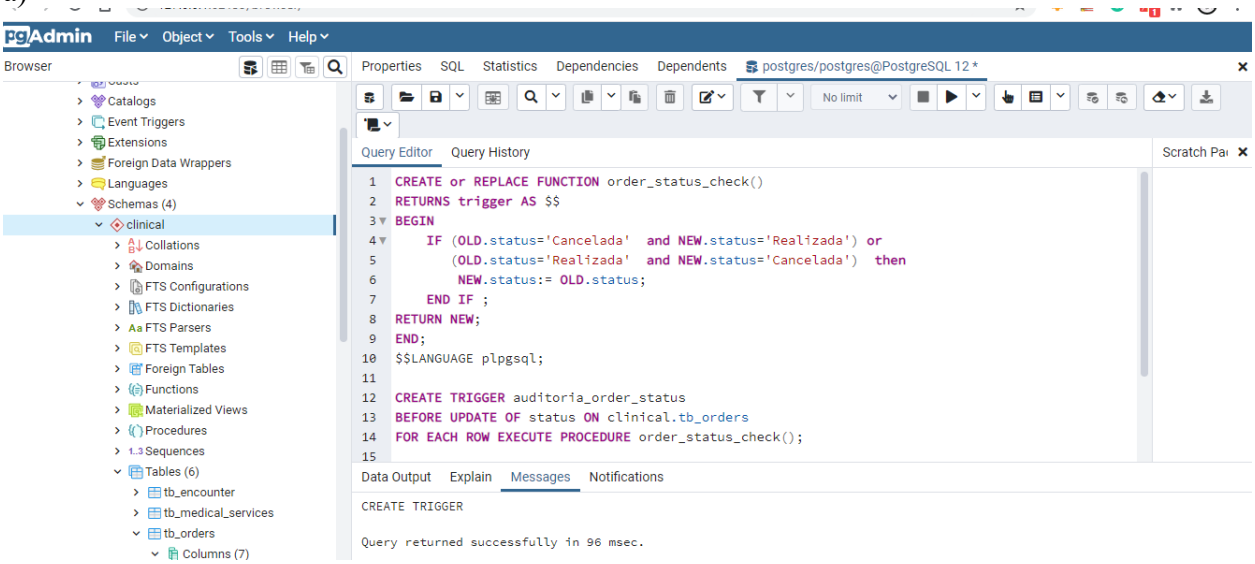


c)

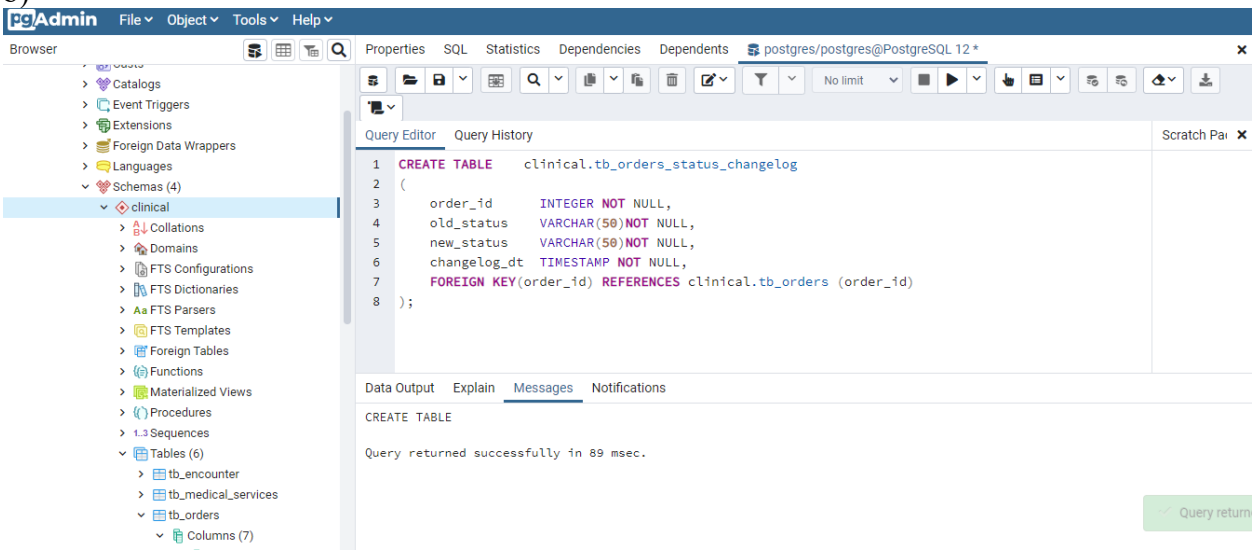

127.0.0.1:62159/browser/
pgAdmin
File
Object
Tools
Help
Browser
Properties
SQL
Statistics
Dependencies
Dependents
postgres/postgres@PostgreSQL 12 *
Servers (1)
PostgreSQL 12
Databases (1)
postgres
Casts
Catalogs
Event Trigger
Extensions
Foreign Data
Languages
Schemas (4)
clinical
No limit
postgres/postgres@PostgreSQL 12
Query Editor
Query History
Scratch Pad
1 CREATE or REPLACE FUNCTION summary_orders()
2 RETURNS SETOF tipo_datos_report AS $$
3 DECLARE
4 datos_clientes tipo_datos_report;
5 prestaciones_year INTEGER;
6 prestaciones_cat_year INTEGER;
7 BEGIN
8 FOR datos_clientes IN SELECT date_part('year',o.created_dt) year_cat, order_code,0
9 FROM clinical.tb_orders o
10 GROUP BY year_cat, order_code
11 ORDER BY year_cat, order_code ASC LOOP
12 prestaciones_cat_year := (SELECT catalog_yearly_orders(datos_clientes.year_cat,datos_clientes.order_cat));
13 prestaciones_year := (SELECT yearly_orders(datos_clientes.year_cat));
14 datos_clientes.percentage := ROUND(prestaciones_cat_year * 100.0 / prestaciones_year, 1);
15
Data Output
Explain
Messages
Notifications
year_cat integer
order_cat integer
percentage double precision
1
2009
1454
28.6

EJERCICIO 3 (30%)

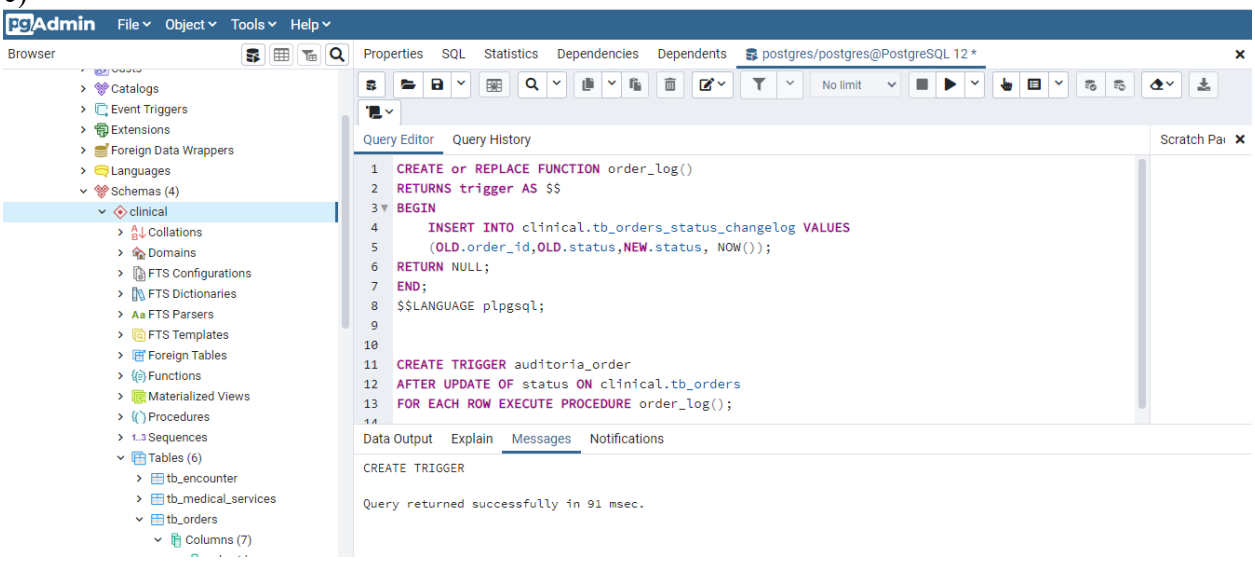
a)



b)



c)


pgAdmin
File
Object
Tools
Help
Browser
Properties
SQL
Statistics
Dependencies
Dependents
postgres/postgres@PostgreSQL 12 *
No limit
Query Editor
Query History
Scratch Pa
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas (4)
clinical
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (6)
tb_encounter
tb_medical_services
tb_orders
Columns (7)
1 CREATE or REPLACE FUNCTION order_log()
2 RETURNS trigger AS $$
3 BEGIN
4 INSERT INTO clinical.tb_orders_status_changelog VALUES
5 (OLD.order_id,OLD.status,NEW.status, NOW());
6 RETURN NULL;
7 END;
8 $$LANGUAGE plpgsql;
9
10
11 CREATE TRIGGER auditoria_order
12 AFTER UPDATE OF status ON clinical.tb_orders
13 FOR EACH ROW EXECUTE PROCEDURE order_log();
Data Output
Explain
Messages
Notifications
CREATE TRIGGER
Query returned successfully in 91 msec.

EJERCICIO 4 (10%)

Cada función tiene una volátil clasificación , la función por defecto es volátil si no se especifica lo contrario en el CREATE FUNCTION, esta característica de las funciones es utilizada por el QUERY-PLANNER de postgreSQL para manejar las consultas asumiendo ciertas características de las funciones de tal manera que pueda optimizar las llamadas a dichas funciones. Los supuesto que realiza sobre las funciones son :

La función es **inmutable** : Tiene como características principales :

- No puede modificar DB
- No puede leer de la DB
- Si tenemos entradas similares(argumentos) debe retorno la misma salida(output)
- Como ejemplo podríamos tener la función lower

La función es **estable** : Tiene como características principales :

- No puede modificar DB
- Puede leer de la DB(cuando el resultado de una función depende de los datos de alguna tabla
- Si tenemos entradas similares(argumentos) debe retorno la misma salida(output)
- Como ejemplo podríamos tener la función : CURRENT_TIMESTAMP o NOW

La función es **volátil** :Es el default y tiene como característica principales :

- Puede leer y modificar DB.
- Puede retornar diferentes datos en cada llamada.
- Puede ser usado como trigger function
- Como ejemplo podríamos tener las funciones : random, nextval o curval.

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 (PEC3).

La fecha límite para entregar la PEC3 es el **09/05/2021**.

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.