Universitat Oberta de Catalunya uoc.edu

Uso de bases de datos

Práctica 2: El lenguaje SQL II

Uso de IA

En esta actividad **no está permitido el uso de herramientas de inteligencia artificial**. En el plan docente y en la web sobre integridad académica y plagio de la UOC encontraréis información sobre qué se considera conducta irregular en la evaluación y las consecuencias que puede tener.

Enunciado

En la primera parte de la práctica hemos realizado consultas y modificaciones sobre los datos mediante el uso de SQL. En esta segunda parte, sobre la misma base de datos, añadiremos lógica utilizando procedimientos almacenados y disparadores.

Para la correcta ejecución de la segunda parte de la práctica, **es necesario volver a crear la base de datos de nuevo e insertar otra vez los datos iniciales utilizando los *scripts* que se proporcionan en este enunciado** (*create_db.sql* e *inserts_db.sql*, respectivamente). Este paso es necesario a causa de la introducción del siguiente cambio en el esquema de la base de datos respecto a la primera parte de la práctica:

- Creación de una nueva tabla llamada *REPORT_WINE*, con las siguientes columnas:
  - *wine_id*: Identificador del vino.
  - *wine_name*: Nombre del vino.
  - *alcohol_content*: Graduación del vino.
  - *category*: Categoria del vino.
  - *price*: Precio del vino.
  - *prizes*: Número de premios que ha recibido el vino.
  - *total_sold*: Cantidad total de cajas vendidas.
  - *orders*: Número de pedidos en los que se ha solicitado el vino.
  - *customer_id*: Identificador del cliente que ha realizado más pedidos del vino.
  - *customer_name*: Nombre del cliente que ha realizado más pedidos del vino.

Nota importante: El SQL implementado en PostgreSQL puede aceptar diferentes variantes de sintaxis, que además pueden diferir según la versión que instaléis, y que pueden ser o no SQL estándar. Evitad (salvo que se indique lo contrario) utilizar sentencias de este tipo, y concentraros en las que se explican en los módulos didácticos. Si usáis sentencias SQL estándar vuestro código funcionará en cualquier SGBD.

Pregunta 1 (15 % puntuación)

Enunciado

Se pide crear un procedimiento almacenado que, dado el identificador de un vino, nos dé algunos datos en referencia de éste. En concreto, queremos su identificador (*wine_id*), el nombre del vino (*wine_name*), la graduación (*alcohol_content*), su categoría (*category*), el precio (*price*), la cantidad de premios que han recibido (*prizes*), la cantidad total de cajas vendidas (*total_sold*), el número de pedidos en los que se ha solicitado (*orders*), y el cliente que ha realizado más pedidos del vino (*customer_id* y *customer_name*). En caso de empate, se debe seleccionar el cliente con más cajas compradas, y si todavía hay empate, el cliente cuyo nombre sea primero en orden alfabético.

Toda esta información la queremos almacenar en la tabla *REPORT_WINE*. La tabla en cuestión se habrá creado con la ejecución del fichero *create_db.sql*, **el cual tendréis que ejecutar antes que nada**. Si ya existen filas en la tabla *REPORT_WINE* para el vino, esta tabla se tendrá que actualizar con los nuevos valores. Además de guardar los datos en la tabla *REPORT_WINE*, el procedimiento devolverá el resultado del *report*.

Habrá que informar al usuario con un mensaje específico cuando no exista ningún vino con el nombre que se ha indicado. También hay que informar cuando el vino nunca ha sido solicitado en un pedido.

La signatura del procedimiento solicitado y el tipo que devolverá son los siguientes:

```
CREATE OR REPLACE FUNCTION update_report_wine(p_wine_id INT)

RETURNS REPORT_WINE_TYPE AS $$
```

donde *REPORT_WINE_TYPE* es de tipo:

```
CREATE TYPE REPORT_WINE_TYPE AS (
      t_wine_id INTEGER,
      t_wine_name VARCHAR(100),
      t_alcohol_content DECIMAL(4,2),
      t_category VARCHAR(50),
      t_price DECIMAL(8,2),
      t_prizes INTEGER,
      t_total_sold INTEGER,
      t_orders INTEGER,
      t_customer_id INTEGER,
      t_customer_name  VARCHAR(100));
```

Nota: Adicionalmente, en el siguiente enlace, encontraréis información sobre errores y mensajes en PL/PostgreSQL: PostgreSQL: Documentation: 16: 43.9. Errors and Messages.

Criterios de evaluación

- Las propuestas de solución que no se puedan ejecutar, es decir, que den error de sintaxis, no serán evaluadas.
- Se valorará positivamente el uso de sentencias SQL estándar, al margen de otros elementos que se puedan indicar en el enunciado.
- Para obtener la máxima nota, el código SQL de vuestra solución tiene que ser eficiente. Por ejemplo, se valorará negativamente hacer más *joins* de las necesarias.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir pruebas que cubran todas las posibles situaciones descritas en el enunciado. Por ejemplo, hay que cubrir todas las posibles situaciones de error.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir los resultados, mediante el uso de pantallazos u otro mecanismo similar.

Respuesta:

```
SET search_path TO ubd_20241;
--Ejercicio 1
CREATE OR REPLACE FUNCTION update_report_wine(p_wine_id INT)
RETURNS REPORT_WINE_TYPE AS $$
DECLARE
   v_report REPORT_WINE_TYPE;  -- Variable para almacenar el tipo compuesto
   v_existe_en_report_wine INT;       -- Variable para comprobar si existe el vino en la tabla
BEGIN


        with estadistica_vino as (
                select  -- estadisticas sobre el vino
                        wine_id
                        , sum(quantity) as total_sold  -- cajas vendidas
                        , count(distinct(order_id)) as orders   -- pedidos
                from order_line
                group by wine_id
        ), mejor_cliente_aux as (
                /*
                y el cliente que ha realizado más pedidos del vino
                (customer_id y customer_name). En caso de empate, se debe seleccionar el cliente con más cajas
                compradas, y si todavía hay empate, el cliente cuyo nombre sea primero en orden alfabético.
                 * */
                select
                        pedido.wine_id
```

```
                        , cliente.customer_id
                        , cliente.customer_name
                        , sum(pedido.quantity) as cajas
                        , count(distinct(pedido.order_id)) as pedidos
                        , rank() over( partition by pedido.wine_id order by sum(pedido.quantity) desc, count(distinct(pedido.order_id)) desc, cliente.customer_name asc ) as ranking
                from order_line pedido
                join customer_order pedido_cliente on ( pedido.order_id = pedido_cliente.order_id )
                join customer cliente on ( pedido_cliente.customer_id = cliente.customer_id )
                group by cliente.customer_id
                                , cliente.customer_name
                                , pedido.wine_id
        ), mejor_cliente as (
                select * from mejor_cliente_aux
                where ranking = 1
        )
        select  -- Info de los vinos
        into v_report
                wine.wine_id
                , wine.wine_name
                , wine.alcohol_content
                , wine.category
                , wine.price
                , wine.prizes
                , stats.total_sold
                , stats.orders
                , cli.customer_id
                , cli.customer_name
        from wine
        left join estadistica_vino stats on ( wine.wine_id = stats.wine_id )
        left join mejor_cliente cli on ( wine.wine_id = cli.wine_id )
        where 1 = 1
                and wine.wine_id = p_wine_id;



        -- Validar si no existen registros
   IF v_report IS NULL THEN
      RAISE WARNING 'No se encontraron registros para el vino con ID %', p_wine_id;
        return NULL;
   END IF;

  IF v_report.t_orders is null then
                RAISE INFO 'El vino con ID % no ha recibido pedidos aún', p_wine_id;
  end if;
```

```
	-- Comprobar si hay registro en la tabla de report_wine
	select
	into v_existe_en_report_wine
		count(1) as existe_en_report_wine
	from report_wine
	where wine_id = p_wine_id;


	if v_existe_en_report_wine > 0 then
		-- UPDATE EN LA TABLA
		update report_wine set
			wine_name = v_report.t_wine_name
			, alcohol_content = v_report.t_alcohol_content
			, category = v_report.t_category
			, price = v_report.t_price
			, prizes = v_report.t_prizes
			, total_sold = v_report.t_total_sold
			, orders = v_report.t_orders
			, customer_id = v_report.t_customer_id
			, customer_name = v_report.t_customer_name
		where wine_id = p_wine_id;

		RAISE INFO 'Actualizado el vino % en la tabla report_wine', p_wine_id;

	else
		-- INSERT EN LA TABLA
		insert into report_wine ( wine_id, wine_name, alcohol_content, category, price, prizes,
total_sold, orders, customer_id, customer_name )
		values ( v_report.t_wine_id, v_report.t_wine_name, v_report.t_alcohol_content,
v_report.t_category
			, v_report.t_price, v_report.t_prizes, v_report.t_total_sold, v_report.t_orders,
v_report.t_customer_id
			, v_report.t_customer_name );

		RAISE INFO 'Insertado el vino % en la tabla report_wine', p_wine_id;
	end if;

	return v_report;

END;
$$ LANGUAGE plpgsql;


-- select update_report_wine(2)
```

Pruebas:

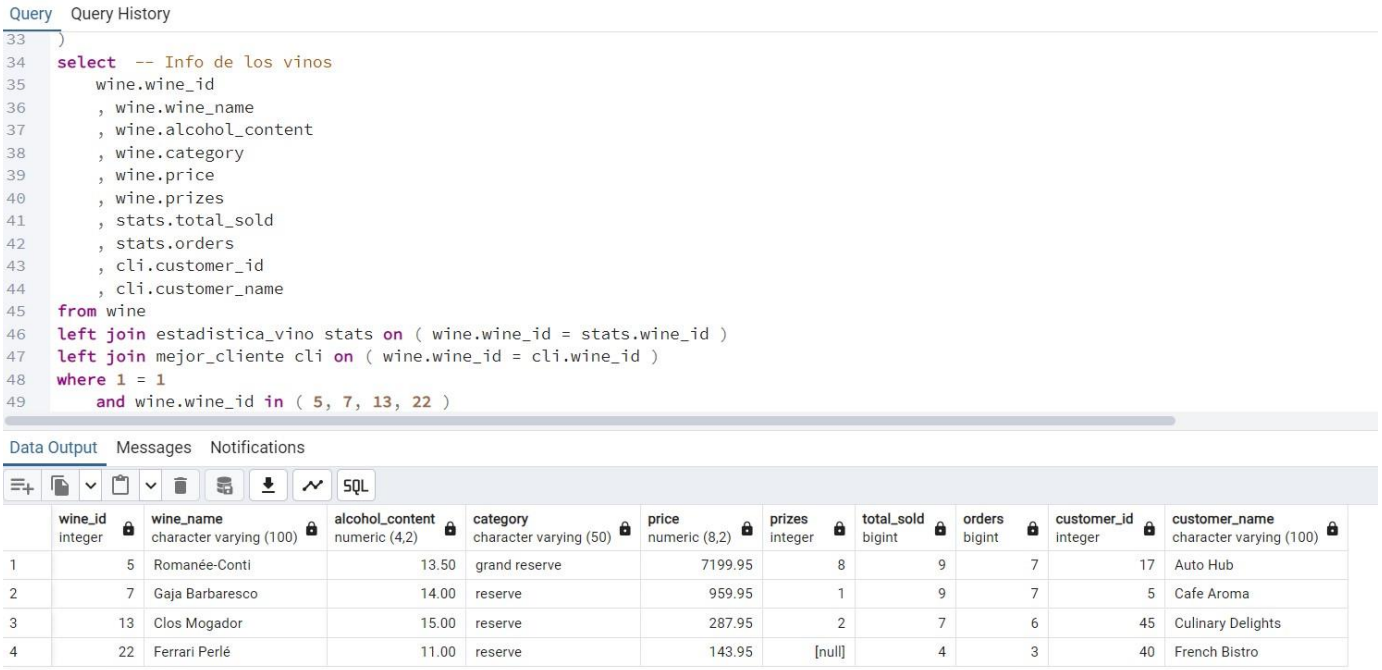
- Obtener listado de vinos aleatorios para comprobar que la información es correcta.

Query:

```
with estadistica_vino as (
	select  -- estadisticas sobre el vino
		wine_id
		, sum(quantity) as total_sold  -- cajas vendidas
		, count(distinct(order_id)) as orders   -- pedidos
	from order_line
	group by wine_id
), mejor_cliente_aux as (
	/*
	y el cliente que ha realizado más pedidos del vino
	(customer_id y customer_name). En caso de empate, se debe seleccionar el cliente con más cajas
	compradas, y si todavía hay empate, el cliente cuyo nombre sea primero en orden alfabético.
	 * */
	select
		pedido.wine_id
		, cliente.customer_id
		, cliente.customer_name
		, sum(pedido.quantity) as cajas
		, count(distinct(pedido.order_id)) as pedidos
		, rank() over( partition by pedido.wine_id order by sum(pedido.quantity) desc, count(distinct(pedido.order_id)) desc, cliente.customer_name asc ) as ranking
	from order_line pedido
	join customer_order pedido_cliente on ( pedido.order_id = pedido_cliente.order_id )
	join customer cliente on ( pedido_cliente.customer_id = cliente.customer_id )
	group by cliente.customer_id
			, cliente.customer_name
			, pedido.wine_id
), mejor_cliente as (
	select * from mejor_cliente_aux
	where ranking = 1
)
select  -- Info de los vinos
	wine.wine_id
	, wine.wine_name
	, wine.alcohol_content
	, wine.category
	, wine.price
	, wine.prizes
	, stats.total_sold
	, stats.orders
	, cli.customer_id
	, cli.customer_name
from wine
left join estadistica_vino stats on ( wine.wine_id = stats.wine_id )
left join mejor_cliente cli on ( wine.wine_id = cli.wine_id )
```

```
where 1 = 1
        and wine.wine_id in ( 5, 7, 13, 22 )
```

Captura:



Query | Query History

```
33  )
34  select  -- Info de los vinos
35      wine.wine_id
36      , wine.wine_name
37      , wine.alcohol_content
38      , wine.category
39      , wine.price
40      , wine.prizes
41      , stats.total_sold
42      , stats.orders
43      , cli.customer_id
44      , cli.customer_name
45  from wine
46  left join estadistica_vino stats on ( wine.wine_id = stats.wine_id )
47  left join mejor_cliente cli on ( wine.wine_id = cli.wine_id )
48  where 1 = 1
49      and wine.wine_id in ( 5, 7, 13, 22 )
```

Data Output | Messages | Notifications

	wine_id integer	wine_name character varying (100)	alcohol_content numeric (4,2)	category character varying (50)	price numeric (8,2)	prizes integer	total_sold bigint	orders bigint	customer_id integer	customer_name character varying (100)
1	5	Romanée-Conti	13.50	grand reserve	7199.95	8	9	7	17	Auto Hub
2	7	Gaja Barbaresco	14.00	reserve	959.95	1	9	7	5	Cafe Aroma
3	13	Clos Mogador	15.00	reserve	287.95	2	7	6	45	Culinary Delights
4	22	Ferrari Perlé	11.00	reserve	143.95	[null]	4	3	40	French Bistro

- Comprobar que los clientes que aparecen para esos vinos, son los que tienen que salir

Query:

```
select
        pedido.wine_id
        , cliente.customer_id
        , cliente.customer_name
        , sum(pedido.quantity) as cajas
        , count(distinct(pedido.order_id)) as pedidos
        , rank() over( partition by pedido.wine_id order by sum(pedido.quantity) desc, count(distinct(pedido.order_id)) desc, cliente.customer_name asc ) as ranking
from order_line pedido
join customer_order pedido_cliente on ( pedido.order_id = pedido_cliente.order_id )
join customer cliente on ( pedido_cliente.customer_id = cliente.customer_id )
where pedido.wine_id in ( 5, 7, 13, 22 )
group by cliente.customer_id
                , cliente.customer_name
                , pedido.wine_id
```

Data Output | Messages | Notifications

	wine_id integer	customer_id integer	customer_name character varying (100)	cajas bigint	pedidos bigint	ranking bigint
1	5	17	Auto Hub	3	2	1
2	5	23	Fashion Forward	2	1	2
3	5	45	Culinary Delights	1	1	3
4	5	42	Portuguese Delicacies	1	1	4
5	5	1	SuperMart	1	1	5
6	5	30	Taste of Italy	1	1	6
7	7	5	Cafe Aroma	2	1	1
8	7	1	SuperMart	2	1	2
9	7	36	Culinary Treasures	1	1	3
10	7	22	Gadget Galaxy	1	1	4
11	7	28	Gourmet Delights	1	1	5
12	7	33	Portuguese Flavors	1	1	6
13	7	26	Tech Trends	1	1	7
14	13	45	Culinary Delights	2	1	1
15	13	10	Book Haven	1	1	2
16	13	46	Food Haven	1	1	3
17	13	34	Mediterranean Bites	1	1	4
18	13	19	Office Outlet	1	1	5
19	13	42	Portuguese Delicacies	1	1	6
20	22	40	French Bistro	2	2	1
21	22	15	Music Mania	2	1	2

- Comprobar que el procedimiento funciona bien
  - Ejecutar procedimiento con un ID que no existe

Query | Query History

```
75      -- Comprobar si hay registro en la tabla de report_wine
76      select
77      into v_existe_en_report_wine
78          count(1) as existe_en_report_wine
79      from report_wine
80      where wine_id = p_wine_id;
81
82      select * from report_wine
83
84      select update_report_wine(500)
85
```

Data Output | Messages | Notifications

```
WARNING:  No se encontraron registros para el vino con ID 500

Successfully run. Total query runtime: 69 msec.
1 rows affected.
```

  - Ejecutar procedimiento con un ID que si existe y no tiene registro en la tabla report_wine

```
3
4       select update_report_wine(20)
5
6
```

ata Output | Messages | Notifications

```
NFO:  Actualizado el vino 20 en la tabla report_wine

uccessfully run. Total query runtime: 76 msec.
 rows affected.
```

- Ejecutar procedimiento con el mismo ID y comprobar que hace un update

Query | Query History

```
76  select
77  into v_existe_en_report_wine
78      count(1) as existe_en_report_wine
79  from report_wine
80  where wine_id = p_wine_id;
81
82  select * from report_wine
83
84  where wine_id = 20
85
86
```

Data Output | Messages | Notifications

	wine_id [PK] smallint	wine_name character varying (255)	alcohol_content numeric (4,2)	category character varying (255)	price numeric (7,2)	prizes integer	total_sold integer	orders integer	customer_id smallint	customer_name character varying (255)
1	20	Masi Amarone	15.00	reserve	239.95	4	7	5	5	Cafe Aroma

- Modificar manualmente el registro

Query | Query History

```
82  select * from report_wine
83
84  where wine_id = 20
85
86  update report_wine set
87          price = 60000
88      where wine_id = 20
89
90
91
92
```

Data Output | Messages | Notifications

	wine_id [PK] smallint	wine_name character varying (255)	alcohol_content numeric (4,2)	category character varying (255)	price numeric (7,2)	prizes integer	total_sold integer	orders integer	customer_id smallint	customer_name character varying (255)
1	20	Masi Amarone	15.00	reserve	60000.00	4	7	5	5	Cafe Aroma

```
update report_wine set
 price = 60000
where wine_id = 20
```

- Ejecutar el procedimiento y comprobar que se actualiza

Query | Query History

```
82  select * from report_wine
83
84  where wine_id = 20
85
86  update report_wine set
87          price = 60000
88      where wine_id = 20
89
90  select update_report_wine(20)
91
92
```

Data Output | Messages | Notifications

```
INFO:  Actualizado el vino 20 en la tabla report_wine

Successfully run. Total query runtime: 96 msec.
1 rows affected.
```

Pregunta 2 (18% puntuación)

Enunciado

En la tabla *WINE* tenemos la columna *stock* con el objetivo de almacenar **el número de cajas disponibles para cada vino.**

Cread un disparador o disparadores, sobre la tabla o tablas que sean necesarias, de forma que se mantengan correctamente actualizada la columna *stock* de la tabla *WINE*, de forma que mantenga el inventario actualizado en tiempo real en función de los pedidos de los clientes. Será necesario informar al usuario con un mensaje específico cuando el *stock* sea insuficiente para un pedido.

En concreto, queremos que esta columna siempre refleje los valores actualizados en función de los cambios.

Podemos suponer que los usuarios o programas nunca actualizarán directamente la columna *stock* de la tabla *WINE*.

Criterios de evaluación

- Las propuestas de solución que no se puedan ejecutar, las que den error de sintaxis, no serán evaluadas.
- Se valorará positivamente el uso de sentencias SQL estándar, al margen de otros elementos que se puedan indicar en el enunciado.
- Para obtener la máxima nota, el código SQL de vuestra solución tiene que ser eficiente. Por ejemplo, se valorará negativamente hacer más *joins* de las necesarias.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir pruebas que cubran todas las posibles situaciones descritas en el enunciado.
- Para obtener la máxima nota, la propuesta de solución tiene que incluir los resultados, mediante pantallazos o de alguna forma similar.

Respuesta:

```
--Actualizar el stock de la tabla vinos en cada insert o actualización en la tabla order_line
create or replace function actualizar_stock()
returns trigger AS $$
        declare v_cant_stock int;
        declare v_cantidad_antigua int;
        declare v_nueva_cantidad int;
begin

        raise info 'Disparador: %', tg_op;


        -- Si es negativo, lanzar error
        if new.quantity < 0 then
```

```
        raise exception 'La cantidad no puede ser inferior a 0';
    end if;

    -- Obtener el stock del vino
    select
    into v_cant_stock
        stock
    from wine
    where wine_id = new.wine_id;

    v_cantidad_antigua = 0;
    if tg_op = 'UPDATE' then
        v_cantidad_antigua = old.quantity;
    end if;


    v_nueva_cantidad = new.quantity - v_cantidad_antigua;

    -- Si no hay stock suficiente cuando es insert, lanzar error
    if  v_cant_stock < v_nueva_cantidad then
        raise exception 'La cantidad del pedido es superior al stock actual del vino %. Cantidad
de pedido: %; Cantidad de stock: %', new.wine_id, v_nueva_cantidad, v_cant_stock;
    end if;


    RAISE INFO 'Actualizando el stock del vino % [ % ] -> [ % ]', new.wine_id, v_cantidad_antigua ,
v_nueva_cantidad;

    -- Si no se ha lanzado error todavía, actualizar el stock
    update wine set
        stock = stock - v_nueva_cantidad
    where wine_id = new.wine_id;


    RETURN NEW;
END;
$$ LANGUAGE plpgsql;




CREATE or replace TRIGGER trigger_wine_stock_actualizar
BEFORE insert OR update
ON order_line
FOR EACH ROW
EXECUTE FUNCTION actualizar_stock();
```

Pruebas:

- Comprobar el stock de vino 1

```
33
34    select
35     wine_id
36        , stock
37    from wine
38    where wine_id = 1;
39
```

Data Output | Messages | Notifications

	wine_id [PK] integer	stock integer
1	1	83

- **Insertar un pedido nuevo para el vino 1**

```
41    insert into order_line (order_id, order_line_id, wine_id, quantity)
42    values (1, 4, 1, 5)
```

Data Output | Messages | Notifications

```
INFO:  Disparador: INSERT
INFO:  Actualizando el stock del vino 1 [ 0 ] -> [ 5 ]
INSERT 0 1

Query returned successfully in 100 msec.
```

```
41    select * from order_line
42    where order_id = 1
43    and order_line_id = 4
44
45
46    insert into order_line (order_id, order_line_id
```

Data Output | Messages | Notifications

	order_id [PK] integer	order_line_id [PK] integer	wine_id integer	quantity integer	discount integer
1	1	4	1	5	[null]

- **Comprobar que se ha disminuido el stock al vino 1**

```
36    select
37     wine_id
38        , stock
39    from wine
40    where wine_id = 1;
41
```

Data Output | Messages | Notifications

	wine_id [PK] integer	stock integer
1	1	78

- Insertar en otro pedido una linea con cantidad de 200. Debe dar error

```
47    insert into order_line (order_id, order_line_id, wine_id, quantity)
48    values (1, 4, 1, 200)
```

Data Output | Messages | Notifications

```
INFO:  Disparador: INSERT

ERROR:  La cantidad del pedido es superior al stock actual del vino 1. Cantidad de pedido: 200; Cantidad de stock: 78
CONTEXT:  función PL/pgSQL actualizar_stock() en la línea 31 en RAISE

SQL state: P0001
```

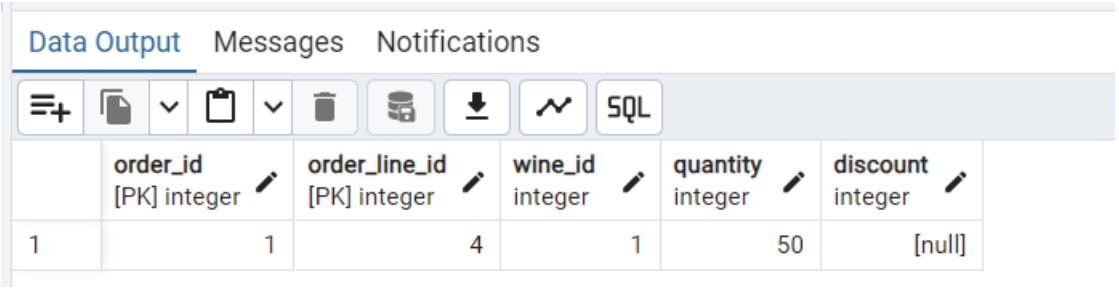
- **Actualizar el pedido 1, asignar a la linea que sea la cantidad de 50. Hay que restarle 45 al stock del vino.**

```
4     update order_line set
5         quantity =50
6     where order_id = 1
7     and order_line_id = 4
8
9
```

)ata Output | Messages | Notifications

```
INFO:  Disparador: UPDATE
INFO:  Actualizando el stock del vino 1 [ 5 ] -> [ 45 ]
JPDATE 1

)uery returned successfully in 73 msec.
```



- **Comprobar que el stock ha actualizado**