EXERCISI 1

Els script del fitxer tenen la forma:

```
DROP TABLE IF EXISTS sale.tb_client CASCADE;

CREATE TABLE sale.tb_client (
	client_code		char(5) not null,
	client_name		varchar(40) not null,
	address			varchar(140),
	city			varchar(25),
	country			varchar(60) not null,
	contact_email		varchar(100),
	phone			varchar(15),
	parent_client_code	char(5),
	created_by_user		varchar(10) not null default 'OS_SYSTEM',
	created_date		date,
	updated_date		date,
	CONSTRAINT tb_client_pk PRIMARY KEY(client_code)
);

ALTER TABLE sale.tb_client ADD CONSTRAINT tb_client_client_fk FOREIGN KEY (parent_client_code)
REFERENCES sale.tb_client (client_code);

CREATE INDEX tb_client_parent_client_code_idx ON sale.tb_client (parent_client_code);
```

Com a detalls:

1. Eliminem la taula si existeix. D'aquesta forma se pot garantir que s'executa el fitxer sencer. Utilitzem el CASCADE per a evitar haver de posar la creació en ordre.
2. Especifiquem l'esquema tant a l'eliminació de la taula com a la creació.
3. Definim totes les constraints (excepte les FK) i els valors per defecte en el moment de creació de la taula.
4. Creem posteriorment les FK, per tal d'evitar haver de fer la creació en ordre.
5. Les cadenes de texte son CHAR (CHARACTER) quan tenen una longitud fixa. VARCHAR quan és variable.
6. Les dates són DATE, que no inclou hora, minut i segon.
7. He intentat que els camps numèrics siguin lo més adequat possible als requeriments del problema.

EXERCISI 2

Hem creat les consultes així com les he entès. Com a detalls:

- Hem utilitzat les OUTER JOIN en els casos en els que entenc que era necessari. Evidentment, si no és necessari, s'haurien de llevar. Milloraria el rendiment.
- Hem evitat utilitzar DISTINCT. Crec que no és necessari en cap cas.
- Hem prioritzat l'ús de WITH.
- No he pogut evitar l'ús de NOT EXISTS.

EXERCISI 3

Hem creat les SQLs demandes. Simplement dir que al final executem un COMMIT per a garantir la persistència. També dir que es pot fer el DELETE del registre "Pavo" sense més perquè no hi ha cap registre fill. Si hagués estat així, hauríem de prendre una decisió (posar a null, eliminar el registre, ...).

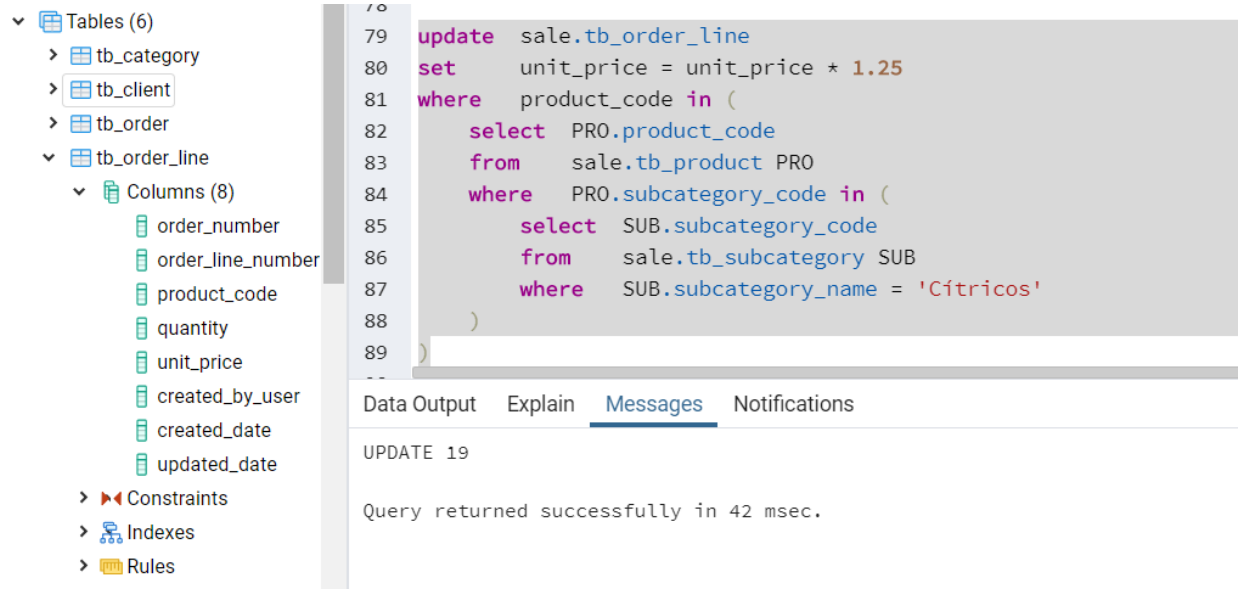
Captures de pantalla:

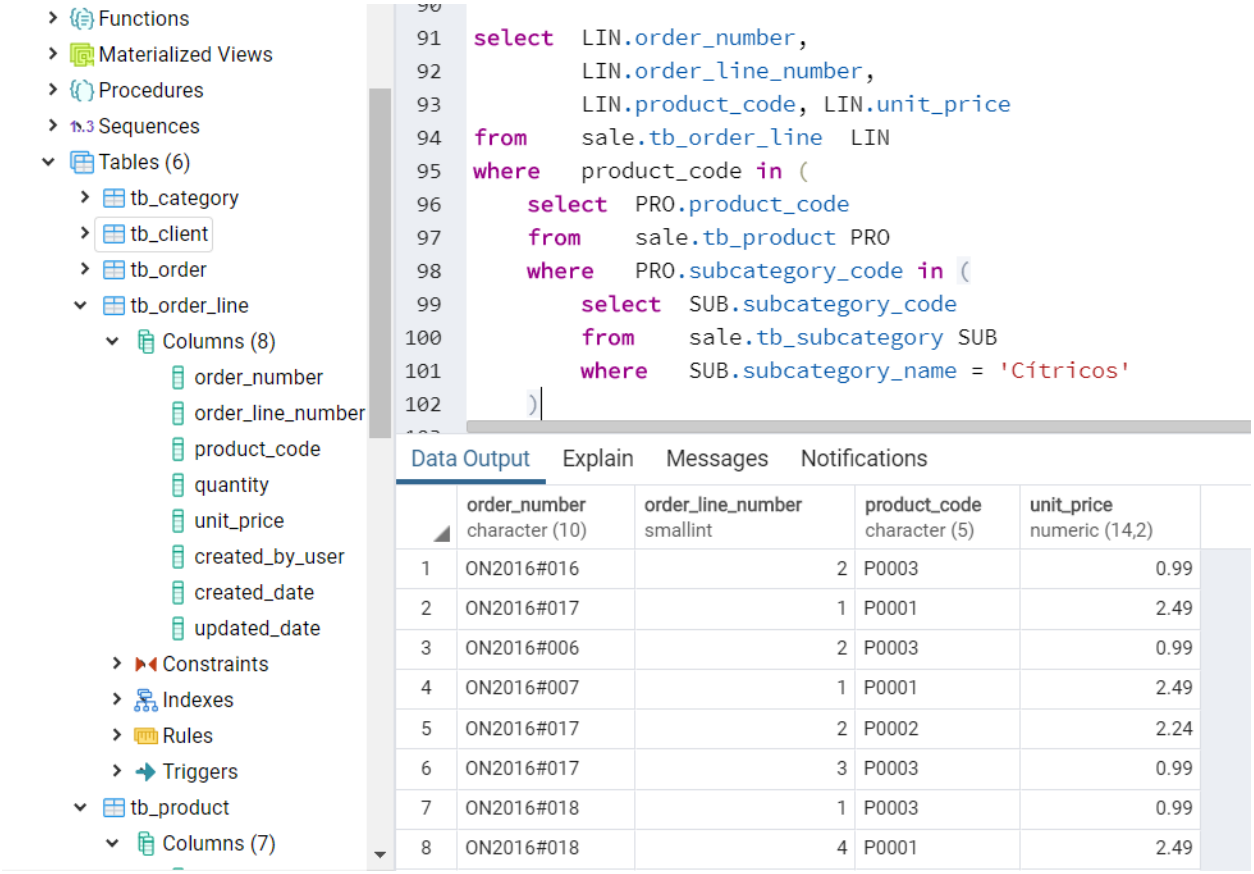
Clients creats

Data Output | Explain | Messages | Notifications

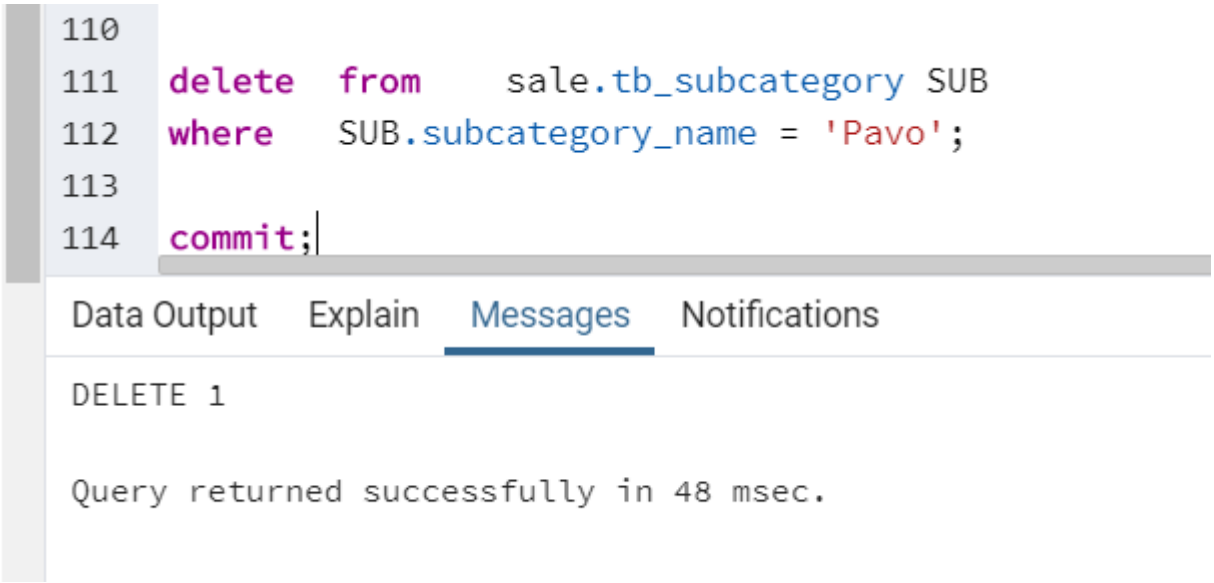
	client_code character (5)	client_name character varying (40)	address character varying (140)	city character varying (25)	country character varying (60)	contact_email character varying (100)	phone character varying (15)	parent_client_code character (5)	created_by_user character varying (10)	crea date
1	C0030	Lidl	Drungüerkanster 38	Frankfurt	Germany	info@lidl.de	[null]	[null]	UOC-Alumno	2017
2	C0031	Lidl - Frankfurt	Drungüerkanster 128	Frankfurt	Germany	frankfurt@lidl.de	998858896	C0030	UOC-Alumno	2017
3	C0032	Lidl – Frankfurt GmB	Drungüerkanster 128	Frankfurt	Germany	frankfurt@lidl.de	998858896	C0031	UOC-Alumno	2017

Canvi de preu de les línees de cítricts





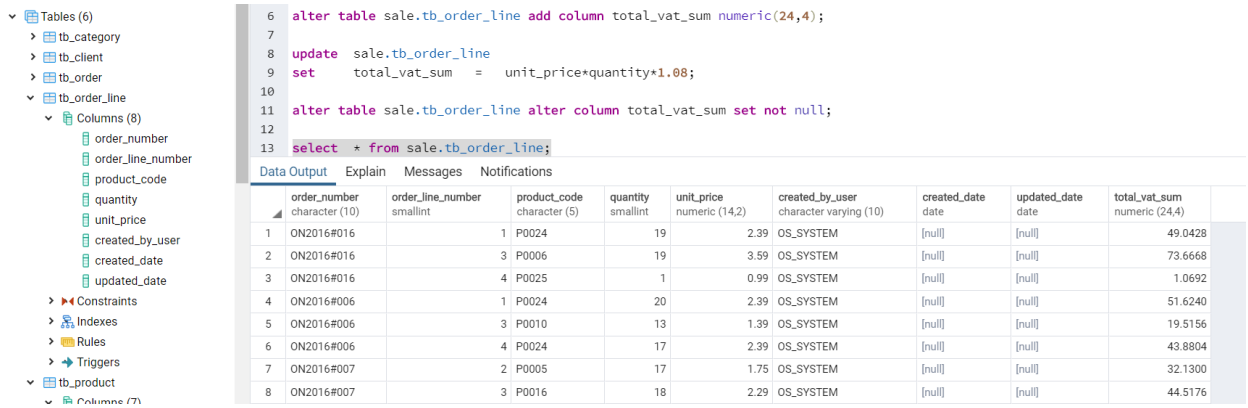
Eliminar categoría Pavo (No ha estat necessari possar a null els registres depenents, ja que no n'hi ha)



EXERCISI 4

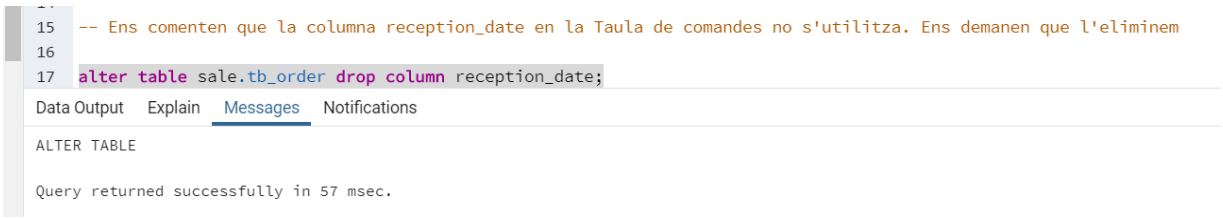
Hem creat les SQLs demandes. Simplement dir que no llencem cap commit perquè la DDL fa un commit implícit.

Captures de pantalla:

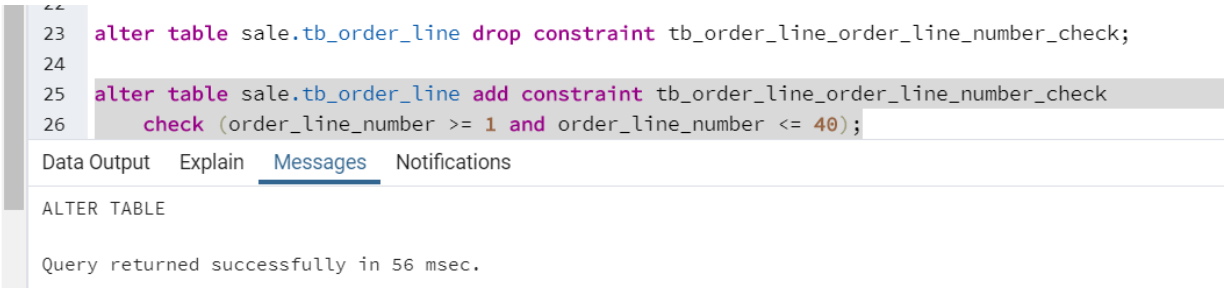


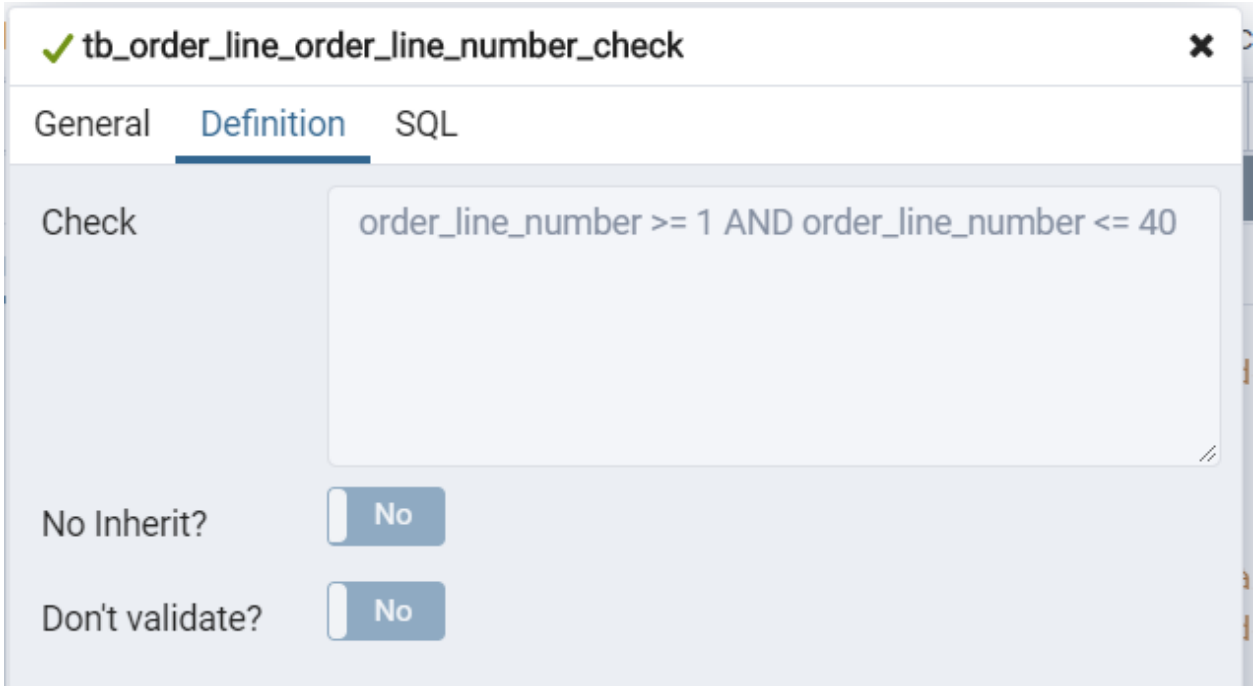
	order_number character (10)	order_line_number smallint	product_code character (5)	quantity smallint	unit_price numeric (14,2)	created_by_user character varying (10)	created_date date	updated_date date	total_vat_sum numeric (24,4)
1	ON2016#016	1	P0024	19	2.39	OS_SYSTEM	[null]	[null]	49.0428
2	ON2016#016	3	P0006	19	3.59	OS_SYSTEM	[null]	[null]	73.6668
3	ON2016#016	4	P0025	1	0.99	OS_SYSTEM	[null]	[null]	1.0692
4	ON2016#006	1	P0024	20	2.39	OS_SYSTEM	[null]	[null]	51.6240
5	ON2016#006	3	P0010	13	1.39	OS_SYSTEM	[null]	[null]	19.5156
6	ON2016#006	4	P0024	17	2.39	OS_SYSTEM	[null]	[null]	43.8804
7	ON2016#007	2	P0005	17	1.75	OS_SYSTEM	[null]	[null]	32.1300
8	ON2016#007	3	P0016	18	2.29	OS_SYSTEM	[null]	[null]	44.5176

Eliminació de la columna



Modificar el check de validació de número de línia:




tb_order_line_order_line_number_check
General
Definition
SQL
Check
order_line_number >= 1 AND order_line_number <= 40
No Inherit?
No
Don't validate?
No

EXERCISI 5

PREPARE

Totes les SQL que se llencen sobre la base de dades han de ser prèviament parsetjades i analitzades. El parseitg inclou validar que totes les taules de la consulta existeixen i són accessibles, que els camps existeixen, que la query està ben formada, etc. L'anàlisi inclou determinar quin és el millor pla d'execució: utilitzar o no un índex, si s'ha de crear una taula temporal en memòria, etc.

Evidentment, aquestes dues operacions tarden un temps. Si una consulta es llença moltes vegades, tot el temps invertit a partir de la segona vegada és temps que ens podríem estalviar si guardem el resultat d'aquestes dues passes.

La instrucció PREPARE fa exactament això, a més de re-escriure la consulta si és optimitzable. Se pot utilitzar per a les comandes SELECT i per a les DML (INSERT, UPDATE i DELETE).

És molt interessant que pot acceptar paràmetres. D'aquesta forma, la SQL no ha de ser exactament igual i tot continua funcionant. Exemples de PREPARE:

Per a un INSERT:

```
PREPARE fooplan (int, text, bool, numeric) AS
    INSERT INTO foo VALUES($1, $2, $3, $4);
EXECUTE fooplan(1, 'Hunter Valley', 't', 200.00);
```

Per a una SELECT:

```
PREPARE usrrptplan (int) AS
    SELECT * FROM users u, logs l WHERE u.usrid=$1 AND
u.usrid=l.usrid
    AND l.date = $2;
EXECUTE usrrptplan(1, current_date);
```

Els paràmetres es quadren per odre i s'especifiquen a la consulta i tenen la forma $1, $2, etc. Si no s'especifica el tipus (exemple del segon paràmetre de la SELECT), s'infereix de la pròpia base de dades.

Font:

https://www.postgresql.org/docs/11/sql-prepare.html

ON-COMMIT

Les taules temporals són taules on les dades poden persistir a dos nivells:

- Sessió. Quan es crea la sessió, la taula és buida. S'insereixen, eliminen, modifiquen i consulten dades independentment de les altres sessions. No es comparteixen les dades entre sessions. Quan la sessió finalitza, les dades desapareixen.
- Transacció. L'aïllament entre sessions és exactament el mateix que a nivell de sessió. La diferència és que cada vegada que una transacció finalitza (ja sigui amb un commit o amb un rollback), la taula es buida.

La clàusula ON COMMIT permet definir, a nivell de taula, si la persistència es produeix a nivell de sessió o de transacció. S'especifica quan es crea la taula i després es pot modificar mitjançant un alter.

Les opcions són:

- PRESERVE ROWS. Valor per defecte. Els registres no s'eliminen quan es fa un COMMIT o un ROLLBACK. Persistència a nivell de sessió.
- DELETE ROWS. Els registres s'eliminen quan es fa un COMMIT o un ROLLBACK. Persistència a nivell de transacció.
- DROP. No només s'eliminen els registres, sinó que s'elimina la taula sencera quan es fa un COMMIT o ROLLBACK. Persistència a nivell de transacció.

Exemple:

```
CREATE [ GLOBAL|LOCAL ] {TEMPORARY | TEMP } TABLE table_name
( column_name data_type,
              ...
                       ...
                                ... ) ON COMMIT DELETE ROWS;
```

Com a nota, dir que GLOBAL|LOCAL permet definir si la taula serà visible des de qualsevol sessió o només des de la sessió actual.

Font:

https://www.postgresql.org/docs/11/sql-createtable.html