Introduction to Databases

PR1: Creating and working with a relational database

Name	Daniel
Surnames	Maestre Sánchez
UOC Username	dmaestresan

Content

Exercise 1

In the create_db.sql file, we provide all the SQL sentences needed to create WINE, ZONE, GRAPE_VARIETY, WINE_GRAPE, CUSTOMER_ORDER, ORDER_LINE and CUSTOMER tables. Therefore, after one last analysis, it is clear that the following enhancements must be implemented:

1. What are the necessary SQL statements to **create** the *WINERY* table according to the definition provided?

```sql
CREATE TABLE WINERY (
    winery_id SERIAL PRIMARY KEY,
    winery_name VARCHAR(100) NOT NULL,
    town VARCHAR(100) NOT NULL,
    established_year INT,
    winery_phone VARCHAR(20),
    sales_representative VARCHAR(100) NOT NULL
);
```

2. You also need to provide the alter SQL statements to allow the following modifications to the database:

   2.1. Table *WINE* needs the following modifications (2%):

   a) Add the *prizes* attribute to store the number of prizes won by the wine. It can take *NULL* as a value.

```sql
--a)
--alter add colum price of win
ALTER TABLE WINE
ADD COLUMN prizes INT;
```

   b) The *alchohol_content* attribute must be between 10º and 11º for white wines (*white*), between 11º and 15º for rosé wines (*rosé*) and, between 13º and 18º for red wines (*red*).

```sql
--b)
-- Remove existing restriction chk_alcohol_content
ALTER TABLE WINE
DROP CONSTRAINT IF EXISTS chk_alcohol_content;

-- Defining the alcohol content by type of wine
ALTER TABLE WINE
ADD CONSTRAINT chk_alcohol_content CHECK (
    (color = 'white' AND alcohol_content BETWEEN 10 AND 11) OR
    (color = 'rosé' AND alcohol_content BETWEEN 11 AND 15) OR
    (color = 'red' AND alcohol_content BETWEEN 13 AND 18)
);
```

   c) The winery identifier attribute (*winery_id*) is a foreign key to *WINERY*.

```sql
--c)
-- Define winery_id in WINE as foreign key to WINERY
ALTER TABLE WINE
ADD CONSTRAINT fk_winery FOREIGN KEY (winery_id) REFERENCES WINERY(winery_id);
```

2.2. Table *CUSTOMER_ORDER* needs the following modifications (2%):

a) Add the *order_amount* attribute to store the total amount of the order. This value must be equal or greater than 0.

```sql
--a)
--Add column for total order amount
ALTER TABLE CUSTOMER_ORDER
ADD COLUMN order_amount DECIMAL(10, 2) CHECK (order_amount >= 0);
```

b) Add the order_reference attribute to store an order identifier for the client reference. This attribute must be a character string and must have a format according to this pattern: XXX-NN-XXXX, where X must be capital letters and N must be numbers.

```sql
--b)
-- Add order reference column with specific formatting XXX-NN-XXX//X-Capital leters N-> Numbers
ALTER TABLE CUSTOMER_ORDER
ADD COLUMN order_reference VARCHAR(12) CHECK (order_reference SIMILAR TO '[A-Z]{3}-[0-9]{2}-[A-Z]{4}');
```

2.3. Table *ORDER_LINE* needs the following modifications (2%):

a) Add the *discount* attribute to store a discount percentage for each order line. This attribute can be *NULL* in case that there is no discount.

```sql
--EXERCISE 1/2.3
-- Add discount column with a maximum limit of 99.99%.
ALTER TABLE ORDER_LINE
ADD COLUMN discount DECIMAL(5, 2) CHECK (discount >= 0 AND discount <= 99.99);
```

2.4. Table *ORDER_LINE* needs the following modifications (2%):

a) We cannot have capital towns with repeated names.

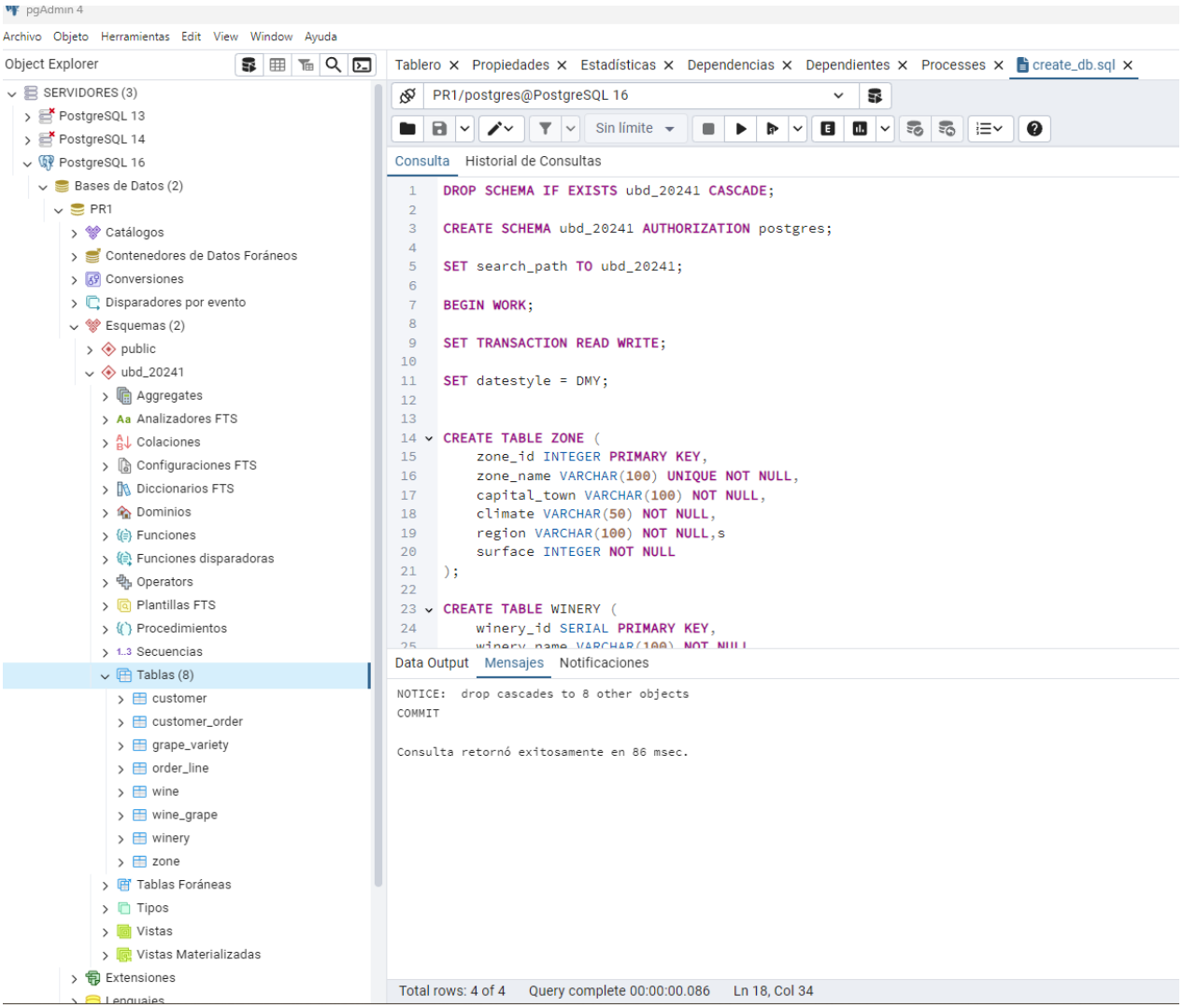
```sql
--a)
-- Avoid duplicate capital names
ALTER TABLE ZONE
ADD CONSTRAINT unique_capital_town UNIQUE (capital_town);
```

b) The *surface* attribute is not necessary and should be removed.

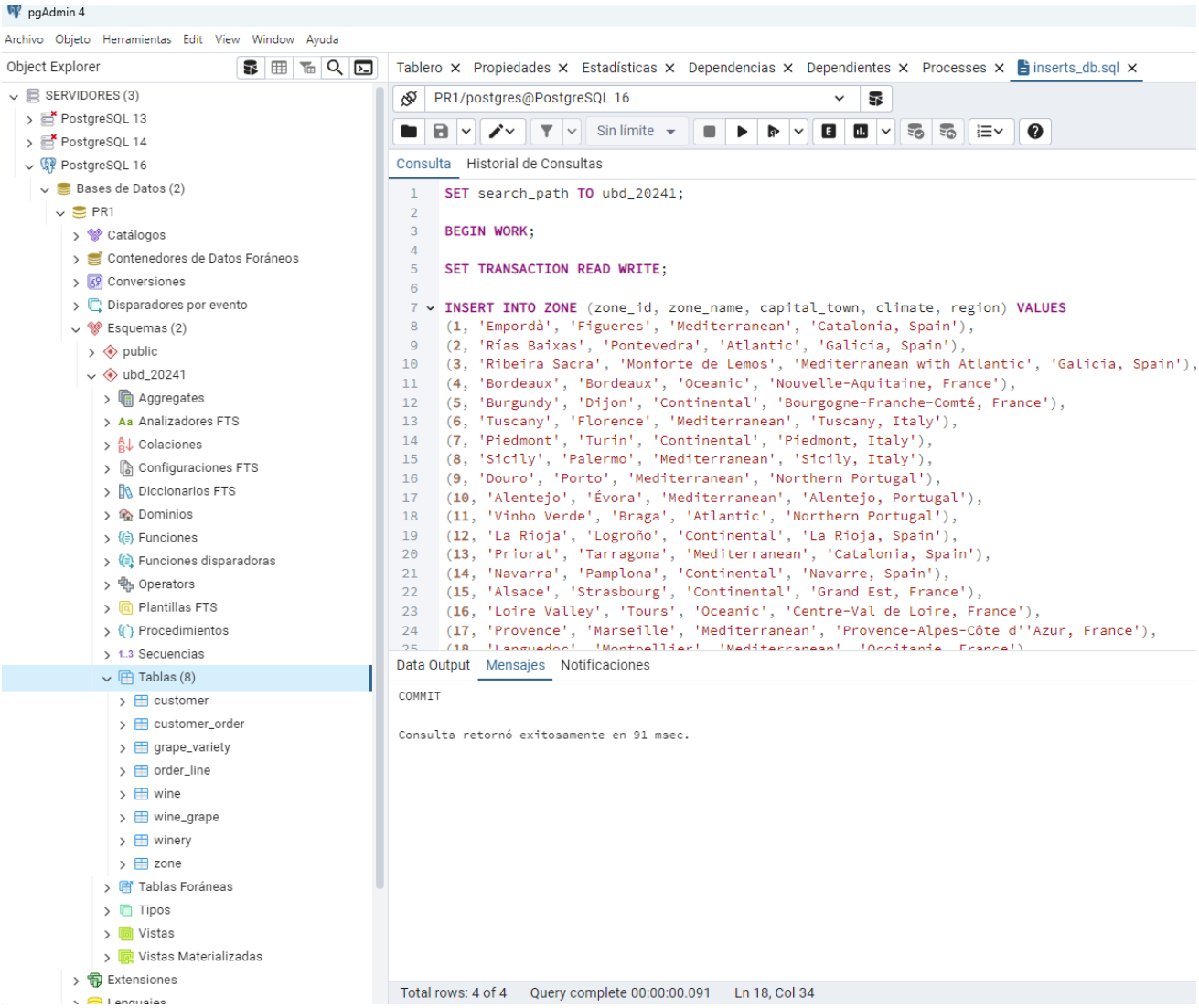
```sql
--b)
-- Eliminate the surface column
ALTER TABLE ZONE
DROP COLUMN surface;
```

Solution final of the system:

Create_db.sql: All OK.



Insert_db.sql: All OK



Exercise 2

1. Write a query that returns the 5 active customers who have placed the most orders. We would like to display the customer code, customer name, total number of orders, and the date of the most recent order. The result of the query should be sorted by the number of orders in descending order, and in case of a tie, by the customer name in alphabetical order.
   SQL:

```sql
SET search_path TO ubd_20241;

SELECT CUSTOMER.customer_id, customer_name, COUNT(CUSTOMER_ORDER.order_id) AS total_orders, MAX(order_date) AS most_recent_order
FROM CUSTOMER_ORDER
JOIN CUSTOMER ON CUSTOMER_ORDER.customer_id = CUSTOMER.customer_id
WHERE customer_status = 'active'
GROUP BY CUSTOMER.customer_id, customer_name
ORDER BY total_orders DESC, customer_name ASC
LIMIT 5;
```

   Result: Data in Excel in the folder Data EX2, in the file 1.csv

Data Output | Mensajes | Notificaciones

	customer_id [PK] integer	customer_name character varying (100)	total_orders bigint	most_recent_order date
1	22	Gadget Galaxy	6	2024-03-01
2	17	Auto Hub	5	2024-01-01
3	33	Portuguese Flavors	4	2024-07-01
4	45	Culinary Delights	3	2023-12-20
5	36	Culinary Treasures	3	2024-08-01

Data Output | Mensajes | Notificaciones

```
Ejecución exitosa. Tiempo de ejecución total de la consulta: 111 msec.
5 filas afectadas.
```

2. Write a query that returns the wines that belong to designations of origin (DOP) that have cataloged at least 5 wines in the database. We would like to display the name of each wine, its category, its color, and the name of the designation of origin (DOP) where it is produced, as well as the total number of wines for each zone. The result of the query should be ordered by the number of wines in descending order, and in case of a tie, by the zone name and the wine name.

SQL:

```sql
SET search_path TO ubd_20241;

SELECT WINE.wine_name, WINE.category, WINE.color,ZONE.zone_name,
    COUNT(WINE.wine_id) OVER (PARTITION BY WINE.zone_id) AS total_wine_in_zone
FROM WINE
JOIN ZONE ON WINE.zone_id = ZONE.zone_id
WHERE WINE.zone_id IN (
        SELECT zone_id
        FROM WINE
        GROUP BY zone_id
        HAVING COUNT(wine_id) >= 5
    )
ORDER BY total_wine_in_zone DESC, ZONE.zone_name, WINE.wine_name;
```

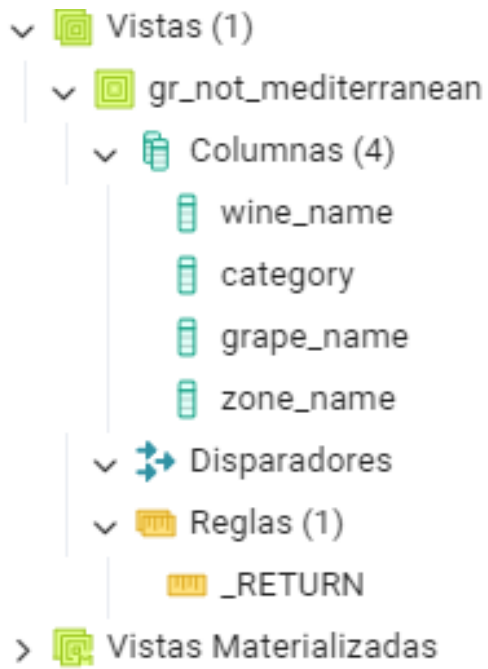
Result: Data in Excel in the folder Data EX2, in the file 2.csv

	wine_name character varying (100)	category character varying (50)	color character varying (20)	zone_name character varying (100)	total_wine_in_zone bigint
1	Aalto PS	reserve	red	Ribera del Duero	11
2	Alión	reserve	red	Ribera del Duero	11
3	Dominio del Águila Reserva	reserve	red	Ribera del Duero	11
4	El Sequé	reserve	red	Ribera del Duero	11
5	Flor de Pingus	reserve	red	Ribera del Duero	11
6	Hacienda Monasterio	reserve	red	Ribera del Duero	11
7	Mauro	reserve	red	Ribera del Duero	11
8	Pago de Carraovejas	reserve	red	Ribera del Duero	11
9	Pingus	grand reserve	red	Ribera del Duero	11
10	Vega Sicilia Único	grand reserve	red	Ribera del Duero	11
11	Vega Sicilia Valbuena	reserve	red	Ribera del Duero	11
12	Artadi Pagos Viejos	grand reserve	red	La Rioja	8
13	Lan Reserva	reserve	red	La Rioja	8
14	López de Heredia Viña Bosconia	reserve	red	La Rioja	8
15	Marqués de Murrieta Reserva	reserve	red	La Rioja	8
16	Marqués de Riscal Gran Reserva	grand reserve	red	La Rioja	8
17	Marqués de Riscal Reserva	reserve	red	La Rioja	8
18	Remelluri Reserva	reserve	red	La Rioja	8
19	Viña Real Crianza	young	red	La Rioja	8

3. Write a view named gr_not_mediterranean that retrieves information about wines that are categorized as ‘grand reserve’ and that do not contain any grape varieties used in wines from designations of origin with a Mediterranean climate (‘Mediterranean’). We would like to display the wine name, its category, the names of the grape varieties used, and the name of the designation of origin. The result of the query should be ordered by the wine name and by the names of the grape varieties.
SQL:

```sql
SET search_path TO ubd_20241;


CREATE VIEW gr_not_mediterranean AS
SELECT WINE.wine_name, WINE.category, GRAPE_VARIETY.grape_name, ZONE.zone_name
FROM WINE
JOIN WINE_GRAPE ON WINE.wine_id = WINE_GRAPE.wine_id
JOIN GRAPE_VARIETY ON WINE_GRAPE.grape_id = GRAPE_VARIETY.grape_id
JOIN ZONE ON WINE.zone_id = ZONE.zone_id
WHERE WINE.category = 'grand reserve' AND ZONE.climate != 'Mediterranean'
ORDER BY WINE.wine_name, GRAPE_VARIETY.grape_name;
```



Result: I have made this programme to see if it gives a good result. Data in Excel in the folder Data EX2, in the file 3.csv

```
SET search_path TO ubd_20241;

SELECT * FROM gr_not_mediterranean;
```

	wine_name character varying (100)	category character varying (50)	grape_name character varying (50)	zone_name character varying (100)
1	Château Margaux	grand reserve	Cabernet Sauvignon	Bordeaux
2	Château Margaux	grand reserve	Merlot	Bordeaux
3	Marqués de Riscal Gran Reserva	grand reserve	Garnacha	La Rioja
4	Marqués de Riscal Gran Reserva	grand reserve	Tempranillo	La Rioja
5	Romanée-Conti	grand reserve	Pinot Noir	Burgundy
6	Vega Sicilia Único	grand reserve	Cabernet Sauvignon	Ribera del Duero
7	Vega Sicilia Único	grand reserve	Tempranillo	Ribera del Duero

Exercise 3

A calculation error has been discovered regarding discounts on pending wine orders (*order_status* equal to '*pending*') where no wine from the designation of origin (DOP) '*Ribera del Duero*' is included, and furthermore, no discount has been applied to any of the items. For this reason, the wine distributor has decided to apply a 10% discount to all lines in the affected orders.

Please propose **a single SQL UPDATE statement** to correct the rows that need to be updated (if more than one statement is used, the task will be considered incorrect). Additionally, once the update has been made, display the set of rows that have been updated.

Important note: when you test the update, take into consideration that you should always start with the same database, that is, you should always have the same initial data set. Otherwise, you may find discrepancies in your analysis.

SQL:

```sql
SET search_path TO ubd_20241;

UPDATE ORDER_LINE
SET discount = 10
WHERE order_id IN (
    SELECT order_id FROM CUSTOMER_ORDER
    WHERE order_status = 'pending'
    AND order_id NOT IN (
        SELECT ORDER_LINE.order_id
        FROM ORDER_LINE
        JOIN WINE ON ORDER_LINE.wine_id = WINE.wine_id
        JOIN ZONE ON WINE.zone_id = ZONE.zone_id
        WHERE ZONE.zone_name = 'Ribera del Duero'
    )
    AND discount IS NULL
);
```

Check: Program to prove if the values was updated

```
SET search_path TO ubd_20241;

SELECT ORDER_LINE.order_id, ORDER_LINE.order_line_id, ORDER_LINE.wine_id, ORDER_LINE.discount,CUSTOMER_ORDER.order_status,
        ZONE.zone_name AS dop_zone_name
FROM ORDER_LINE
JOIN CUSTOMER_ORDER ON ORDER_LINE.order_id = CUSTOMER_ORDER.order_id
JOIN WINE ON ORDER_LINE.wine_id = WINE.wine_id
JOIN ZONE ON WINE.zone_id = ZONE.zone_id
WHERE ORDER_LINE.discount = 10 AND CUSTOMER_ORDER.order_status = 'pending'AND ZONE.zone_name != 'Ribera del Duero'
ORDER BY ORDER_LINE.order_id, ORDER_LINE.order_line_id;
```

Result:

	order_id integer	order_line_id integer	wine_id integer	discount numeric (5,2)	order_status character varying (20)	dop_zone_name character varying (100)
1	74	1	14	10.00	pending	Navarra
2	79	1	8	10.00	pending	Sicily
3	82	1	1	10.00	pending	Empordà
4	88	1	19	10.00	pending	Abruzzo