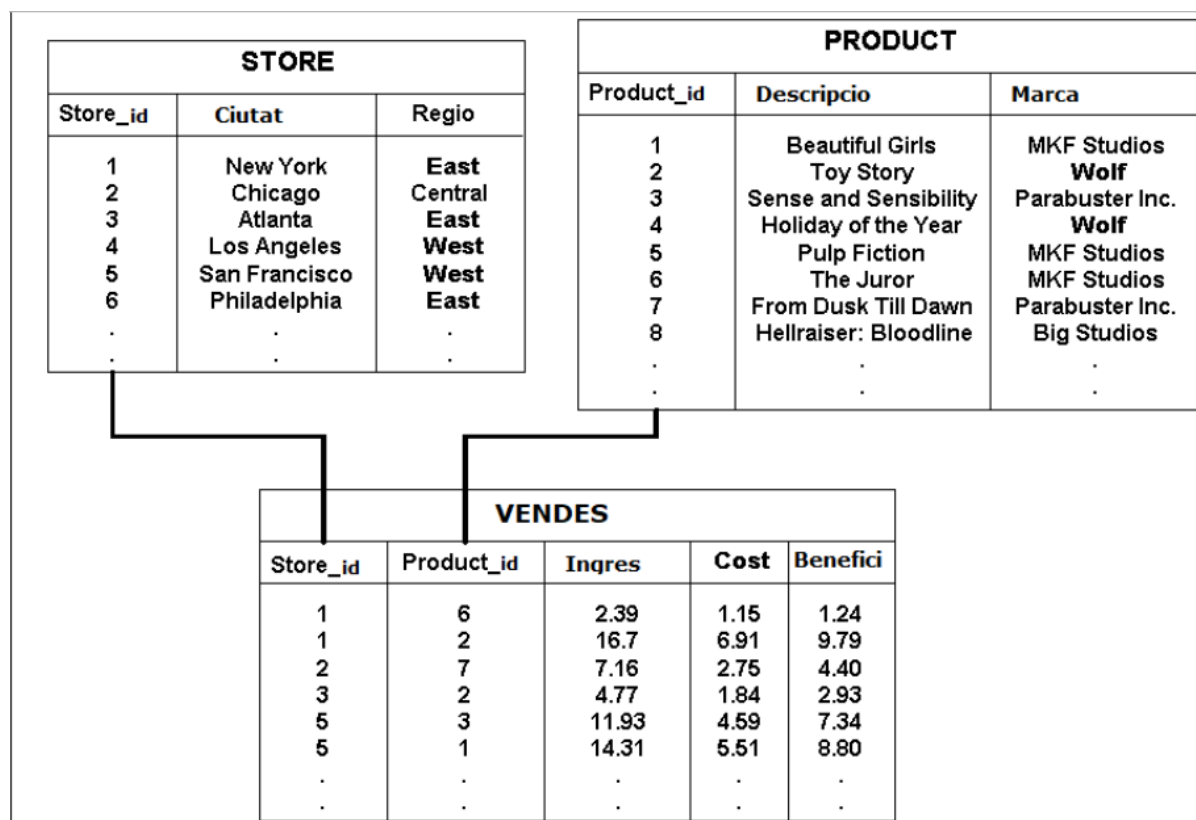


Pràctica 4 - UF2-DDL CREATE TABLE-CREATE VIEW

Exercici 1 - taules

Agafant la base de dades de "botiga" com exemple crear un esquema "store" on hi haurà les taules necessàries per suportar la descripció que es mostra en la següent imatge/diagrama:

- hi ha botigues
- hi ha productes
- hi ha un inventari de vendes, quantitat de productes venuts en cada botiga



1. Identificar la relació de taules de BD a crear.

◦ A banda de les taules que es mostren en la imatge cal pensar en altres possibles taules que es podrien crear per suportar el model, per ex: una taula de ciutats.

2. Identificar les columnes que ha de tenir cada taula de BD i de quin tipus..

- No hi ha una relació directe entre la relació de columnes que es mostra en la imatge i la relació de columnes que ha de tenir una taula de BD..

- A l'hora de definir de quin tipus és cada columna, cal tenir en compte quina funció fa. Per ex: si tenim una taula de CIUTATS, a la taula STORE no hi haurà una columna ciutat on guardar un nom sinó un conlumna ciutat_id on guardar la referència a la taula CIUTATS

3. Identificar la clau primària de cada taula.

4. Identificar les relacions d'integritat referencial entre les taules.

5. Identificar altres possibles restriccions (constraints) que es podrien aplicar a les columnes de les taules.

- ◦Valors null , not null.

- ◦Checks

Una vegada has recollit tota la informació per a crear les taules de la BD. Construeix les sentències de CREATE TABLE corresponents i execute-les. A l'informe de la pràctica s'han de mostrar les sentències de CREATE TABLE que s'han fet, cal que estiguin en format text copiable, i explicar una mica per a que serveix cada taula i les relacions que s'han identificat entre elles.

	Taula ciutats	Taula regio	Taula store	Taula producte	Taula vendes
PRIMARY KEY	ciutat_id (Id únic per cada ciutat)	regio_id (Id únic per cada regio)	store_id (Id únic per cada botiga)	producte_id (Id únic per cada producte)	store_id, producte_id
FOREIGN KEY	regio_id (Referència amb taula regio)		ciutat_id, regio_id (Referències amb les taules ciutats i regio)		store_id, producte_id (Claus externes que fan referències amb store i producte)
Altres columnes	nom_ciutat (Nom de la ciutat)	nom_regio (Nom de la regio)		descripcio (Descripció del producte) marca (La marca del producte)	ingres cost benefici (Xifres de les transaccions)
Funció	Informació sobre les ciutats	Informació sobre les regions	Informació sobre les botigues	Informació sobre els productes disponibles	Registrar les transaccions, incloent la botiga i els productes, amb els seus beneficis, ingressos i costos.

```
DROP DATABASE IF EXISTS store;
CREATE DATABASE store CHARACTER SET utf8mb4;
USE store;

CREATE TABLE regio (
    regio_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nom_regio VARCHAR(50) NOT NULL
);

CREATE TABLE ciutats (
    ciutat_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nom_ciutat VARCHAR(50) NOT NULL,
    regio_id INT UNSIGNED,
    FOREIGN KEY (regio_id) REFERENCES regio(regio_id)
);

CREATE TABLE store (
    store_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ciutat_id INT UNSIGNED,
    regio_id INT UNSIGNED,
    FOREIGN KEY (ciutat_id) REFERENCES ciutats(ciutat_id),
    FOREIGN KEY (regio_id) REFERENCES regio(regio_id)
);

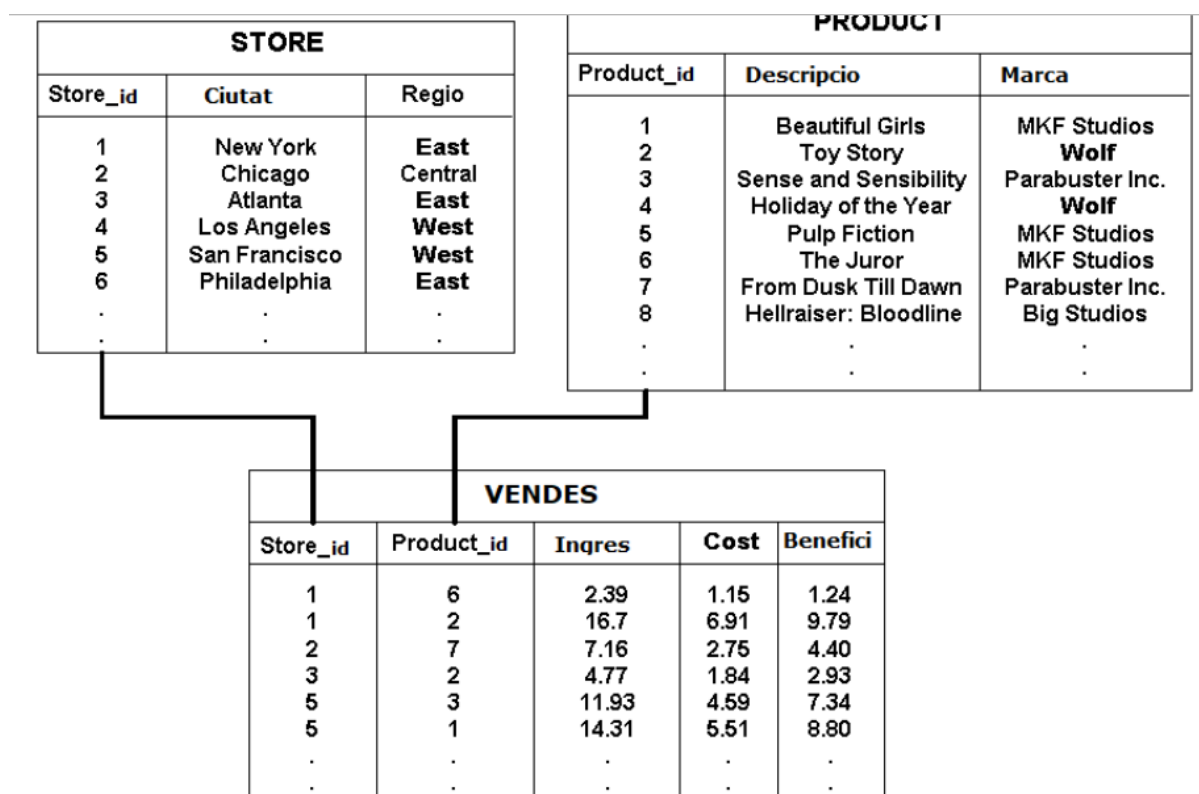
CREATE TABLE producte (
    producte_id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    descripcio VARCHAR(50) NOT NULL,
    marca VARCHAR(50) NOT NULL
);

CREATE TABLE vendes (
    store_id INT UNSIGNED,
    producte_id INT UNSIGNED,
    ingres DECIMAL(10, 2) NOT NULL,
    cost DECIMAL(10, 2) NOT NULL,
    benefici DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (store_id, producte_id),
    FOREIGN KEY (store_id) REFERENCES store(store_id),
    FOREIGN KEY (producte_id) REFERENCES producte(producte_id)
);
```

Exercici 2 - taules

Fent servir les noves taules creades a la BD store, inserir-hi informació per a poder-hi fer consultes. Les dades a inserir no s'han de ser les mateixes que es mostren a la imatge, poden ser altres ciutat, regions

Les tasques a fer són:



1. Select "Store" que retorni la llista de botigues
2. Select "Product" que retorni la llista de productes
3. Select "Vendes" que retorni l'inventari de vendes de cada producte en cada botiga. A l'informe de la pràctica s'han de recollir les sentències SELECT i el resultat, en format text la query i com a taula el resultat.

```
USE store;
```

```
INSERT INTO regio (nom_regio) VALUES
```

```
('Catalunya'),  
( 'Madrid'),  
( 'Andalucía'),  
( 'Galícia'),  
( 'País Basc');
```

```
INSERT INTO ciutats (nom_ciutat, regio_id) VALUES
```

```
('Barcelona', 1),  
( 'Girona', 1),  
( 'Lleida', 1),  
( 'Tarragona', 1),  
( 'Madrid', 2),  
( 'Sevilla', 3),  
( 'Màlaga', 3),  
( 'València', 4),  
( 'Santiago de Compostela', 4),  
( 'Bilbao', 5),  
( 'Donostia', 5);
```

```
INSERT INTO store (ciutat_id, regio_id) VALUES
```

```
(1, 1),  
(2, 1),  
(3, 1),  
(4, 1),  
(5, 2),  
(6, 3),  
(7, 3),  
(8, 4),  
(9, 4),  
(10, 5),  
(11, 5);
```

```
INSERT INTO producte (descripcio, marca) VALUES
```

```
('Portàtil HP', 'HP'),  
( 'Mòbil Samsung', 'Samsung'),  
( 'TV Sony', 'Sony'),  
( 'Impressora Epson', 'Epson'),  
( 'Càmera Canon', 'Canon'),  
( 'Aire condicionat LG', 'LG');
```

```
INSERT INTO vendes (store_id, producte_id, ingres, cost, benefici) VALUES
```

```
(1, 1, 1200.00, 800.00, 400.00),  
(1, 2, 600.00, 400.00, 200.00),  
(2, 1, 1000.00, 700.00, 300.00),  
(3, 3, 1500.00, 1000.00, 500.00),  
(4, 2, 800.00, 500.00, 300.00),  
(5, 3, 1200.00, 900.00, 300.00),  
(6, 1, 800.00, 500.00, 300.00),  
(7, 2, 700.00, 400.00, 300.00),
```

```
(8, 3, 1300.00, 900.00, 400.00),  
(9, 1, 1100.00, 700.00, 400.00),  
(10, 2, 600.00, 300.00, 300.00),  
(11, 3, 1400.00, 1000.00, 400.00);
```

```
SELECT * FROM store;
```

The screenshot shows a SQL IDE interface with three tabs: 'SQL File 11*', 'SQL File 3*', and 'SQL File 4*'. The 'SQL File 4*' tab is active, displaying the following SQL code:

```
1 • USE store;  
2  
3 • SELECT * FROM store;  
4  
5  
6
```

Below the code editor, the 'Result Grid' is visible, showing the results of the query. The grid has three columns: 'store_id', 'ciutat_id', and 'regio_id'. The results are as follows:

	store_id	ciutat_id	regio_id
1	1	1	1
2	2	2	1
3	3	3	1
4	4	4	1
5	5	5	2
6	6	6	3
7	7	7	3
8	8	8	4
9	9	9	4
10	10	10	5
11	11	11	5

Below the result grid, the 'Output' panel is visible, showing the execution log. The log contains two entries:

#	Time	Action	Message
1	00:32:40	USE store	0 row(s) affected
2	00:32:40	SELECT * FROM store LIMIT 0, 1000	11 row(s) returned

```
SELECT * FROM producte;
```

The screenshot shows a SQL IDE interface. At the top, a toolbar includes icons for file operations, a 'Limit to 1000 rows' dropdown, and other utility icons. Below the toolbar, a SQL query is entered in a text area:

```
1 • USE store;  
2  
3 • SELECT * FROM producte;  
4  
5  
6  
7
```

Below the query editor, the 'Result Grid' is displayed, showing a table with three columns: 'producte_id', 'descripcio', and 'marca'. The table contains six rows of data:

producte_id	descripcio	marca
1	Portàtil HP	HP
2	Mòbil Samsung	Samsung
3	TV Sony	Sony
4	Impressora Epson	Epson
5	Càmera Canon	Canon
6	Aire condicionat LG	LG

At the bottom of the IDE, the 'Output' pane shows the 'Action Output' for the executed query:

#	Time	Action	Message
1	00:33:40	USE store	0 row(s) affected
2	00:33:40	SELECT * FROM producte LIMIT 0, 1000	6 row(s) returned

```
SELECT  
  s.store_id,  
  c.nom_ciutat,  
  r.nom_regio,  
  p.producte_id,  
  p.descripcio,  
  v.ingres,  
  v.cost,  
  v.benefici  
FROM vendes v  
JOIN store s ON v.store_id = s.store_id  
JOIN ciutats c ON s.ciutat_id = c.ciutat_id  
JOIN regio r ON s.regio_id = r.regio_id  
JOIN producte p ON v.producte_id = p.producte_id;
```


The screenshot shows a SQL IDE interface. The top pane contains a SQL query with 18 lines of code. The bottom pane displays the results of the query in a table format. The table has 8 columns: store_id, nom_ciutat, nom_regio, producte_id, descripcio, ingres, cost, and benefici. The results show 12 rows of data, including store locations like Barcelona, Girona, Lleida, Tarragona, Madrid, Sevilla, Málaga, València, and Santiago de Compostela, along with product details and financial data.

```
1 • USE store;  
2  
3 • SELECT  
4     s.store_id,  
5     c.nom_ciutat,  
6     r.nom_regio,  
7     p.producte_id,  
8     p.descripcio,  
9     v.ingres,  
10    v.cost,  
11    v.benefici  
12 FROM vendes v  
13 JOIN store s ON v.store_id = s.store_id  
14 JOIN ciutats c ON s.ciutat_id = c.ciutat_id  
15 JOIN regio r ON s.regio_id = r.regio_id  
16 JOIN producte p ON v.producte_id = p.producte_id;  
17  
18
```

	store_id	nom_ciutat	nom_regio	producte_id	descripcio	ingres	cost	benefici
▶	1	Barcelona	Catalunya	1	Portàtil HP	1200.00	800.00	400.00
	1	Barcelona	Catalunya	2	Mòbil Samsung	600.00	400.00	200.00
	2	Girona	Catalunya	1	Portàtil HP	1000.00	700.00	300.00
	3	Lleida	Catalunya	3	TV Sony	1500.00	1000.00	500.00
	4	Tarragona	Catalunya	2	Mòbil Samsung	800.00	500.00	300.00
	5	Madrid	Madrid	3	TV Sony	1200.00	900.00	300.00
	6	Sevilla	Andalucía	1	Portàtil HP	800.00	500.00	300.00
	7	Málaga	Andalucía	2	Mòbil Samsung	700.00	400.00	300.00
	8	València	Galícia	3	TV Sony	1300.00	900.00	400.00
	9	Santiago de Compostela	Galícia	1	Portàtil HP	1100.00	700.00	400.00
10	10	Rilhan	País Basc	2	Mòbil Samsung	600.00	300.00	300.00

Result 3 x

Output

Action Output

#	Time	Action	Message
✓ 1	01:05:10	USE store	0 row(s) affected
✓ 2	01:05:10	SELECT s.store_id, c.nom_ciutat, r.nom_regio, p.producte_id, p.descripcio, v.ingres, v.cost, v.benefici FROM vendes v JOIN store s ...	12 row(s) returned

Exercici 3 - taules

Fes consultes al diccionari de dades, "INFORMATION_SCHEMA", per obtenir informació de la definició de les taules que acabes de crear.

- informació referent a taules
- informació referent a columnes
- informació referent a constraints



Escriu 5 sentències de consultes al diccionari de dades, s'han d'incloure a l'informe de la pràctica la consulta i el resultat, en format text la query i com a taula el resultat.

USE store;

1. Info sobre les taules, com el seu tipus.

```
SELECT table_name, table_type  
FROM information_schema.tables  
WHERE table_schema = 'store';
```

```
1 • USE store;  
2  
3 • SELECT table_name, table_type  
4 FROM information_schema.tables  
5 WHERE table_schema = 'store';  
6  
7  
8  
9  
10  
11
```

Result Grid		
Filter Rows: <input type="text"/>		
Export:  Wrap Cell Contents: 		
	TABLE_NAME	TABLE_TYPE
▶	ciutats	BASE TABLE
	producte	BASE TABLE
	regio	BASE TABLE
	store	BASE TABLE
	vendes	BASE TABLE

tables 1 x		
Output		
Action Output		
#	Time	Action
✓ 1	22:07:22	USE store
✓ 2	22:07:22	SELECT table_name, table_type FROM information_schema.tables WHERE table_schema = 'store' LIMIT 0, 1000

2. Info sobre les columnes, com el seu tipus.

```
SELECT column_name, data_type, is_nullable  
FROM information_schema.columns  
WHERE table_name = 'store' AND table_schema = 'store';
```

```
1 • USE store;  
2  
3 • SELECT column_name, data_type, is_nullable  
4   FROM information_schema.columns  
5   WHERE table_name = 'store' AND table_schema = 'store';  
6  
7  
8  
9  
10  
11  
12
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
▶	store_id	int	NO
	ciutat_id	int	YES
	regio_id	int	YES

columns 2 ×

Output

Action Output ▼

#	Time	Action
✓ 1	22:09:30	USE store
✓ 2	22:09:30	SELECT column_name, data_type, is_nullable FROM information_schema.columns WHE

3. Constraints, com primary key o foreign key.

```
SELECT constraint_name, constraint_type
FROM information_schema.table_constraints
WHERE table_name = 'vendes' AND table_schema = 'store';
```

```
1 • USE store;
2
3 • SELECT constraint_name, constraint_type
4 FROM information_schema.table_constraints
5 WHERE table_name = 'vendes' AND table_schema = 'store';
6
7
8
9
10
11
12
13
```

	CONSTRAINT_NAME	CONSTRAINT_TYPE
PRIMARY	PRIMARY KEY	
vendes_ibfk_1	FOREIGN KEY	
vendes_ibfk_2	FOREIGN KEY	

table_constraints 3 x

Output :

Action Output

	#	Time	Action
✓	1	22:11:00	USE store
✓	2	22:11:00	SELECT constraint_name, constraint_type FROM information_schema.table_constraints W

4. Foreign Keys a la taula vendes

```
1 • USE store;  
2  
3 • SELECT constraint_name, column_name, referenced_table_name, referenced_column_name  
4 FROM information_schema.key_column_usage  
5 WHERE table_name = 'vendes' AND table_schema = 'store' AND referenced_table_name IS NOT NULL;  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

Result Grid				
Filter Rows: <input type="text"/>				
Export: Wrap Cell Content:				
	CONSTRAINT_NAME	COLUMN_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
▶	vendes_ibfk_1	store_id	store	store_id
	vendes_ibfk_2	producte_id	producte	producte_id

key_column_usage 4 x			
Output			
Action Output			
#	Time	Action	
✓ 1	22:15:18	USE store	
✓ 2	22:15:18	SELECT constraint_name, column_name, referenced_table_name, referenced_column_name FROM information_schema.key_column	

5. Primary Keys de la taula vendes

```
SELECT column_name
FROM information_schema.key_column_usage
WHERE table_name = 'vendes' AND table_schema = 'store' AND
constraint_name = 'PRIMARY';
```

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search. The main editor displays the following SQL query:

```
1 • USE store;
2
3 • SELECT column_name
4 FROM information_schema.key_column_usage
5 WHERE table_name = 'vendes' AND table_schema = 'store' AND constraint_name = 'PRIMARY';
6
7
8
9
10
11
12
13
14
15
16
```

Below the editor, the 'Result Grid' section shows the query results:

COLUMN_NAME
store_id
producte_id

The bottom section, titled 'key_column_usage 12', shows the 'Output' of the query execution:

#	Time	Action
✓ 1	22:37:27	USE store
✓ 2	22:37:27	SELECT column_name FROM information_schema.key_column_usage WHERE table_name = 'vendes' AND table_schema = 'store' AND constraint_name = 'PRIMARY';

Exercici 1 - vistes

Fent servir les noves taules creades a l'esquema store, crear una VISTA per poder fer un LLISTAT DE VENDES com es mostra a la imatge:

VENDES				
Store_id	Product_id	Ingres	Cost	Benefici
1	6	2.39	1.15	1.24
1	2	16.7	6.91	9.79
2	7	7.16	2.75	4.40
3	2	4.77	1.84	2.93
5	3	11.93	4.59	7.34
5	1	14.31	5.51	8.80
.
.

LLISTAT VENDES				
Ciutat	Descripció	Ingres	Cost	Benefici
New York	Toy Story	2.39	1.15	1.24
New York	Pulp Fiction	16.7	6.91	9.79
Chicago	Toy Story	7.16	2.75	4.40
Chicago	Pulp Fiction	4.77	1.84	2.93
Chicago	The Juror	11.93	4.59	7.34
Los Angeles	Toy Story	14.31	5.51	8.80
.
.

La vista ha de tenir la llista de columnes necessària per a poder fer consultes del tipus:

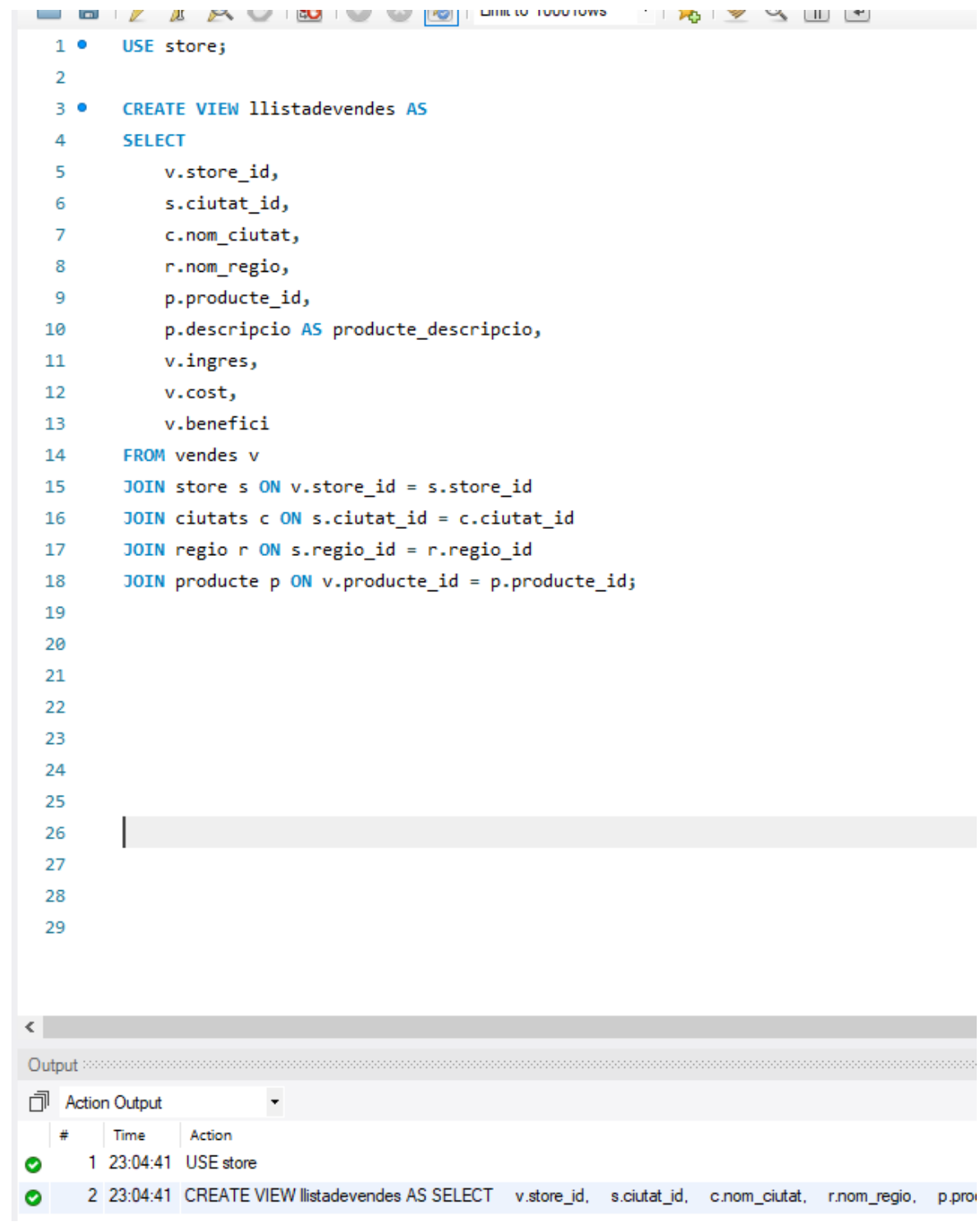
- productes d'una ciutat
- productes d'una regió
- ...

El primer pas a l'hora de crear la vista és transformar l'inventari de vendes, on hi ha codis per la botiga i el producte, en una inventari on en lloc de codis hi ha noms.

1. Crear la vista.

```
CREATE VIEW llistadevendes AS
SELECT
  v.store_id,
  s.ciutat_id,
  c.nom_ciutat,
  r.nom_regio,
  p.producte_id,
  p.descripcio AS producte_descripcio,
  v.ingres,
  v.cost,
  v.benefici
FROM vendes v
JOIN store s ON v.store_id = s.store_id
JOIN ciutats c ON s.ciutat_id = c.ciutat_id
```

JOIN regio r ON s.regio_id = r.regio_id
JOIN producte p ON v.producte_id = p.producte_id;



The screenshot shows a SQL IDE with a script editor and an output window. The script editor contains the following SQL code:

```
1 • USE store;  
2  
3 • CREATE VIEW llistadevendes AS  
4 SELECT  
5     v.store_id,  
6     s.ciutat_id,  
7     c.nom_ciutat,  
8     r.nom_regio,  
9     p.producte_id,  
10    p.descripcio AS producte_descripcio,  
11    v.ingres,  
12    v.cost,  
13    v.benefici  
14 FROM vendes v  
15 JOIN store s ON v.store_id = s.store_id  
16 JOIN ciutats c ON s.ciutat_id = c.ciutat_id  
17 JOIN regio r ON s.regio_id = r.regio_id  
18 JOIN producte p ON v.producte_id = p.producte_id;  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29
```

The output window shows the execution results:

#	Time	Action
1	23:04:41	USE store
2	23:04:41	CREATE VIEW llistadevendes AS SELECT v.store_id, s.ciutat_id, c.nom_ciutat, r.nom_regio, p.pro

2. A partir de la vista fer les següents consultes, (si heu fet servir uns noms de ciutat i regions diferents feu servir els vostres):
- Llistat de productes de "Chicago"


```
SELECT DISTINCT producte_descripcio  
FROM llistadevendes  
WHERE nom_ciutat = 'Barcelona';
```

```
1 • USE store;  
2  
3 • SELECT DISTINCT producte_descripcio  
4 FROM llistadevendes  
5 WHERE nom_ciutat = 'Barcelona';  
6
```

<
Result Grid
Filter Rows: <input type="text"/>
Export
producte_descripcio
▶ Portàtil HP
Mòbil Samsung

l·listadevendes2 x

Output



Action Output

	#	Time	Action
✓	1	23:07:18	USE store
✓	2	23:07:18	SELECT DISTINCT producte_descripcio FR

- Total d'ingressos, costos, beneficis de "Chicago"

```
SELECT
    SUM(ingres) AS total_ingres,
    SUM(cost) AS total_cost,
    SUM(benefici) AS total_beneficis
FROM llistadevendes
WHERE nom_ciutat = 'Barcelona';
```

```
1 • USE store;
2
3 • SELECT
4     SUM(ingres) AS total_ingres,
5     SUM(cost) AS total_cost,
6     SUM(benefici) AS total_beneficis
7 FROM llistadevendes
8 WHERE nom_ciutat = 'Barcelona';
9
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:			
	total_ingres	total_cost	total_beneficis
▶	1800.00	1200.00	600.00

Result 3 x

Output




Action Output

#	Time	Action
✓ 1	23:07:18	USE store
✓ 2	23:07:18	SELECT DISTINCT producte_descripcio FROM llistade
✓ 3	23:08:50	USE store
✓ 4	23:08:50	SELECT SUM(ingres) AS total_ingres, SUM(cost) ,

- Llistat de productes de la regió "Central"

```
SELECT DISTINCT producte_descripcio  
FROM llistadevendes  
WHERE nom_regio = 'Catalunya';
```


```
1 • USE store;  
2  
3 • SELECT DISTINCT producte_descripcio  
4 FROM llistadevendes  
5 WHERE nom_regio = 'Catalunya';  
6  
7
```

< Result Grid  Filter Rows: Export:  Wrap Cell Content: 

	producte_descripcio
▶	Portàtil HP
	Mòbil Samsung
	TV Sony

l·listadevendes4 x

Output


 Action Output

	#	Time	Action
✓	1	23:07:18	USE store
✓	2	23:07:18	SELECT DISTINCT producte_descripcio FROM llistadevendes WHERE nom_ciutat = 'Barcelona'
✓	3	23:08:50	USE store
✓	4	23:08:50	SELECT SUM(ingres) AS total_ingres, SUM(cost) AS total_cost, SUM(benefici) AS total_b
✓	5	23:12:02	USE store
✓	6	23:12:02	SELECT DISTINCT producte_descripcio FROM llistadevendes WHERE nom_regio = 'Catalunya'

- Total d'ingressos, costos, beneficis de la regió "Central"

```
SELECT
    SUM(ingres) AS total_ingres,
    SUM(cost) AS total_cost,
    SUM(benefici) AS total_beneficis
FROM llistadevendes
WHERE nom_regio = 'Catalunya';
```

```
1 • USE store;
2
3 • SELECT
4     SUM(ingres) AS total_ingres,
5     SUM(cost) AS total_cost,
6     SUM(benefici) AS total_beneficis
7 FROM llistadevendes
8 WHERE nom_regio = 'Catalunya';
9
10
11
```

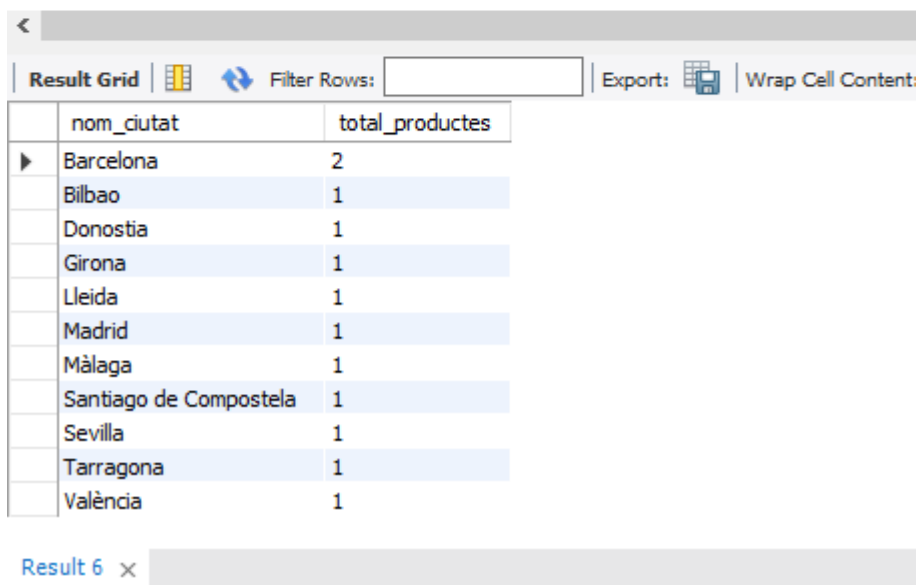
<			
Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Center			
	total_ingres	total_cost	total_beneficis
▶	5100.00	3400.00	1700.00

- Total de productes de cada ciutat.

```
SELECT
    nom_ciutat,
    COUNT(DISTINCT producte_id) AS total_productes
```

FROM llistadevendes
GROUP BY nom_ciutat;

```
1 • USE store;  
2  
3 • SELECT  
4     nom_ciutat,  
5     COUNT(DISTINCT producte_id) AS total_productes  
6 FROM llistadevendes  
7 GROUP BY nom_ciutat;  
8  
9  
10  
11
```



nom_ciutat	total_productes
Barcelona	2
Bilbao	1
Donostia	1
Girona	1
Lleida	1
Madrid	1
Màlaga	1
Santiago de Compostela	1
Sevilla	1
Tarragona	1
València	1

- El producte amb ingressos més elevats de cada ciutat.

```
SELECT  
    nom_ciutat,
```

```
    producte_descripcio,  
    ingres AS max_ingres  
FROM llistadevendes AS lv  
WHERE (nom_ciutat, ingres) IN (  
    SELECT  
        nom_ciutat,  
        MAX(ingres) AS max_ingres  
    FROM llistadevendes  
    GROUP BY nom_ciutat  
);
```

```
1 • USE store;
2
3 • SELECT
4     nom_ciutat,
5     producte_descripcio,
6     ingres AS max_ingres
7 FROM llistadevendes AS lv
8 WHERE (nom_ciutat, ingres) IN (
9     SELECT
10        nom_ciutat,
11        MAX(ingres) AS max_ingres
12    FROM llistadevendes
13    GROUP BY nom_ciutat
14 );
15
16
17
18
```

Result Grid

	nom_ciutat	producte_descripcio	max_ingres
▶	Barcelona	Portàtil HP	1200.00
	Girona	Portàtil HP	1000.00
	Lleida	TV Sony	1500.00
	Tarragona	Mòbil Samsung	800.00
	Madrid	TV Sony	1200.00
	Sevilla	Portàtil HP	800.00
	Màlaga	Mòbil Samsung	700.00
	València	TV Sony	1300.00
	Santiago de Compostela	Portàtil HP	1100.00
	Bilbao	Mòbil Samsung	600.00
	Donostia	TV Sony	1400.00

l·listadevendes7 x

Output

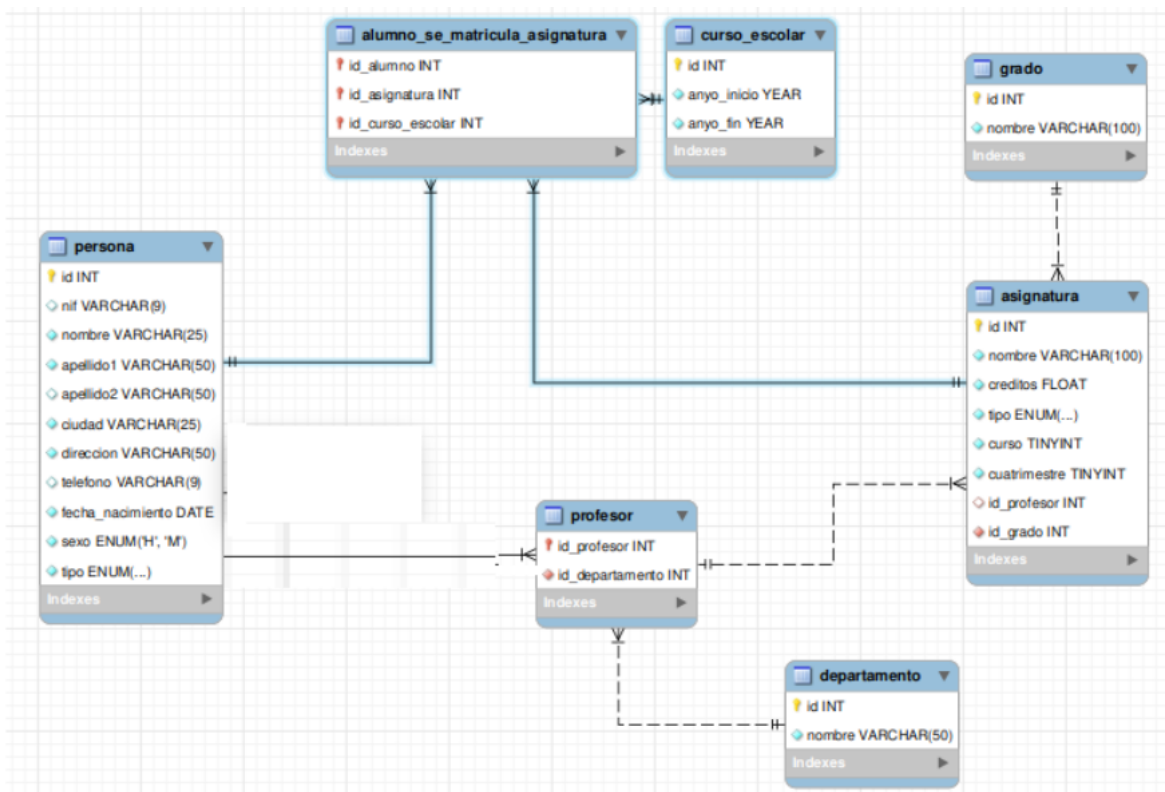
Action Output

#	Time	Action
✓ 1	23:14:08	USE store
✓ 2	23:14:08	SELECT nom_ciutat, producte_descripcio, ingres AS max_ingres FROM llistadevendes AS lv WHERE (no

A l'informe de la pràctica s'ha de recollir la sentència CREATE VIEW, també s'han d'ajuntar evidències de les selects fetes a partir de la vista i del resultat. Comprovar que les selects fetes a partir de la vista donen el mateix resultat que selects sense fer servir la vista.

Exercici 2 - vistes

A partir de l'esquema de la base de dades "universitat".



1. Proposeu 3 vistes per a fer consultes que podrien ser útils en aquesta BD , per exemple: una vista d'assignatures d'un departament.

```
-- vista de professors per departament
CREATE VIEW universitat.v_dept_prof AS
SELECT
    d.id AS id_dept, d.nom AS nom_dept,
    p.id AS id_professor, p.nom AS nom_prof, p.cognom1 AS
cognom1_prof, p.cognom2 AS cognom2_prof, p.sexe AS sexe_prof
FROM
    universitat.departament d
    JOIN universitat.professor pr ON d.id = pr.id_departament
    JOIN universitat.persona p ON pr.id_professor = p.id;

-- vista d'alumnes de cada curs escolar
```



```
CREATE OR REPLACE VIEW universitat.v_alumnes_curs_escolar AS
SELECT
    p.id AS id_alumne,
    p.nom AS nom_alumne,
    p.cognom1 AS cognom1_alumne,
    p.cognom2 AS cognom2_alumne,
    aa.id_grau as id_grau,
    cu.id AS id_curs_escolar,
    cu.any_inici AS any_inici_curs_escolar,
    cu.any_fi AS any_fi_curs_escolar
FROM
    (SELECT DISTINCT id_alumne AS id_alumne, id_curs_escolar AS
id_curs_escolar,
        (select id_grau from universitat.assignatura ass where
ass.id=al.id_assignatura) as id_grau
        FROM universitat.alumne_es_matricula_assignatura al) aa
    JOIN universitat.curs_escolar cu ON aa.id_curs_escolar = cu.id
    JOIN universitat.persona p ON aa.id_alumne = p.id;

-- vista d'alumnes matriculats a cada assignatura
CREATE OR REPLACE VIEW universitat.v_alumnes_assignatures AS
SELECT
    p.id AS id_alumne,
    p.nom AS nom_alumne,
    p.cognom1 AS cognom1_alumne,
    p.cognom2 AS cognom2_alumne,
    ass.id_grau as id_grau,
    ass.id AS id_assignatura,
    ass.nom AS nom_assignatura,
    if(ass.curs=1,'primer',
    if(ass.curs=2,'segon',
        if(ass.curs=3,'tercer',
            if(ass.curs=4,'quart','altres')))) AS curs_assignatura
FROM
    universitat.alumne_es_matricula_assignatura aa
    JOIN universitat.assignatura ass ON aa.id_assignatura = ass.id
    JOIN universitat.persona p ON aa.id_alumne = p.id;
```

2. Creu una vista per a fer inserts a la taula de persones, aquesta vista ha de mostrar les persones que tenen un nif que comenci per '3'.

◦ Quan creem la vista s'ha d'incloure la restricció corresponent per tal que a partir de la vista no es puguin inserir persones que no compleixin la condició esmentada: nif que comenci per '3'.

- Feu un parell de INSERT a partir de la vista, un que sigui OK i un que doni ERROR. A l'informe de la pràctica s'han de mostrar les sentències CREATE VIEW i explicar per què es fan servir.
- Per les select del punt 1, mostrar la query i les files del resultat
- Pels inserts del punt 2, mostrar les sentències i el resultat OK o ERROR.

```
CREATE VIEW vista_insercio_persones AS
SELECT
id,nif,nom,cognom1,cognom2,ciutat,adreca,telefon,data_naixement,sexe,tipus
FROM persona
WHERE
nif LIKE '3%';

ALTER VIEW vista_insercio_persones
AS
SELECT * FROM persona
WHERE nif LIKE '3%' AND tipus IN ('professor', 'alumne');
```

```
1 • use universitat;
2
3 • CREATE VIEW vista_insercio_persones AS
4 SELECT
5 id,nif,nom,cognom1,cognom2,ciutat,adreca,telefon,data_naixement,sexe,tipus
6 FROM persona
7 WHERE
8 nif LIKE '3%';
9
10 • ALTER VIEW vista_insercio_persones
11 AS
12 SELECT * FROM persona
13 WHERE nif LIKE '3%' AND tipus IN ('professor', 'alumne');
14
```

Output

Action Output

#	Time	Action	
✓ 1	23:53:25	use universitat	0
✓ 2	23:53:25	CREATE VIEW vista_insercio_persones AS SELECT id,nif,nom,cognom1,cognom2,ciutat,adreca,telefon,data_naixement,sexe,tipus FROM persona WHE...	0
✓ 3	23:53:25	ALTER VIEW vista_insercio_persones #es fa servir per modificar la definició d'una vista existent. AS SELECT * FROM persona WHERE nif LIKE '3%' AND...	0

Exercici 3 - vistes

Fes consultes al diccionari de dades, "INFORMATION_SCHEMA", per obtenir informació de la definició de les vistes que has creat.

Escriu 3 sentències de consultes al diccionari de dades, s'han d'incloure a l'informe de la pràctica la consulta i el resultat.

1. Info vistes

```
SELECT TABLE_NAME, VIEW_DEFINITION
FROM INFORMATION_SCHEMA.VIEWS
WHERE TABLE_SCHEMA = 'store' AND TABLE_NAME = 'llistadevendes';
```

```
1 • USE store;  
2  
3 • SELECT TABLE_NAME, VIEW_DEFINITION  
4 FROM INFORMATION_SCHEMA.VIEWS  
5 WHERE TABLE_SCHEMA = 'store' AND TABLE_NAME = 'llistadevendes';  
6  
7
```

The screenshot shows a database management tool interface. On the left, a 'Result Grid' displays a table with two columns: 'TABLE_NAME' and 'VIEW_DEFINITION'. The row for 'llistadevendes' is selected, showing its definition. On the right, an 'Edit Data for VIEW_DEFINITION (TEXT)' window shows the full SQL definition of the view, which is a complex join query involving tables like 'vendes', 'store', 'ciutats', 'regio', and 'producte'.

TABLE_NAME	VIEW_DEFINITION
llistadevendes	select 'v'.'store_id' AS 'store_id', 's'.'ciutat_id' AS 'ciutat_id', 'c'.'nom_ciutat' AS 'nom_ciutat', 'r'.'nom_regio' AS 'nom_regio', 'p'.'producte_id' AS 'producte_id', 'p'.'descripcio' AS 'producte_descripcio', 'v'.'ingres' AS 'ingres', 'v'.'cost' AS 'cost', 'v'.'benefici' AS 'benefici' from (((('store'.'vendes' 'v' join 'store'.'store' 's' on (('v'.'store_id' = 's'.'store_id')))) join 'store'.'ciutats' 'c' on (('s'.'ciutat_id' = 'c'.'ciutat_id')))) join 'store'.'regio' 'r' on (('s'.'regio_id' = 'r'.'regio_id')) join 'store'.'producte' 'p' on (('v'.'producte_id' = 'p'.'producte_id'))

2. Columnes de la vista

```
SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH  
FROM INFORMATION_SCHEMA.COLUMNS  
WHERE TABLE_SCHEMA = 'store' AND TABLE_NAME = 'llistadevendes';
```

```
1 • USE store;  
2  
3 • SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH  
4 FROM INFORMATION_SCHEMA.COLUMNS  
5 WHERE TABLE_SCHEMA = 'store' AND TABLE_NAME = 'llistadevendes';  
6  
7  
8
```

<

Result Grid | | Filter Rows: | Export: | Wrap Cell Content:

	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
▶	store_id	int	NULL
	ciutat_id	int	NULL
	nom_ciutat	varchar	50
	nom_regio	varchar	50
	producte_id	int	NULL
	producte_descripcio	varchar	50
	ingres	decimal	NULL
	cost	decimal	NULL
	benefici	decimal	NULL

COLUMNS 9 ×

Output

Action Output ▼

#	Time	Action
✓ 1	23:19:50	USE store
✓ 2	23:19:50	SELECT COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH FROM INFORMATION.

3. Veure les vistes del esquema store

```
SELECT  
    TABLE_NAME,  
    VIEW_DEFINITION  
FROM INFORMATION_SCHEMA.VIEWS  
WHERE TABLE_SCHEMA = 'store';
```

```
1 • USE store;
2
3 • SELECT
4     TABLE_NAME,
5     VIEW_DEFINITION
6 FROM INFORMATION_SCHEMA.VIEWS
7 WHERE TABLE_SCHEMA = 'store';
8
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
TABLE_NAME	VIEW_DEFINITION			
llistadevendes	select `v`.`store_id` AS `store_id`,`s`.`ciuta...			

VIEWS 14 x

Output

Action Output

#	Time	Action
✓ 1	23:22:12	USE store
✓ 2	23:22:12	SELECT TABLE_NAME, VIEW_DEFINITION FROM INFORMATION_SCHEMA.VIEWS WHERE TABLE_SCHE