

## Tasca S3.01. Manipulació de taules

Fernando Poblete

### Descripció

En aquest sprint, es simula una situació empresarial en la qual has de realitzar diverses manipulacions en les taules de la base de dades. Al seu torn, hauràs de treballar amb índexs i vistes. En aquesta activitat, continuaràs treballant amb la base de dades que conté informació d'una empresa dedicada a la venda de productes en línia. En aquesta tasca, començaràs a treballar amb informació relacionada amb targetes de crèdit.

### Nivell 1

#### - Exercici 1

La teva tasca és dissenyar i crear una taula anomenada "credit\_card" que emmagatzemi detalls crucials sobre les targetes de crèdit. La nova taula ha de ser capaç d'identificar de manera única cada targeta i establir una relació adequada amb les altres dues taules ("transaction" i "company"). Després de crear la taula serà necessari que ingressis la informació del document denominat "dades\_introduir\_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.

- Para crear la tabla, ponemos atención al nombre y tipo de datos que queremos que contenga. Lo anterior lo podemos deducir revisando los datos en el archivo 'datos\_introducir\_credit'.

```
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date)
VALUES ('CcU-2938', 'TR301950312213576817638661', '5424465566813633', '3257',
'984', '10/30/22');
```

- Decidimos considerar las variables numéricas con las que no se realizan cálculos, 'pin' y 'cvv', como simples caracteres (VARCHAR), ya que estas pueden comenzar con '0'. El '0' en estas variables es un dígito necesario para la verificación de la transacción, y se omitiría si las variables se consideraran como INT, por ejemplo.

- De manera similar, decidimos cargar las fechas a la tabla 'credit\_card' como VARCHAR ya que el formato de fechas no coincide con el formato estándar de MySQL.

- Usamos el comando CREATE TABLE, especificando que el 'id' en la tabla 'credit\_card' será la llave primaria de esta tabla.

- También incluimos la restricción UNIQUE para el IBAN y el PAN, que son identificadores únicos para cada cuenta bancaria y tarjeta de crédito respectivamente.

The screenshot shows a SQL IDE with a tab labeled 'estructura\_datos\_user\*'. The SQL editor contains the following code:

```

1  /* NIVEL 1 - EJERCICIO 1: Creamos la tabla credit_card */
2  CREATE TABLE IF NOT EXISTS credit_card (
3      id VARCHAR(20) PRIMARY KEY,
4      iban VARCHAR(50) UNIQUE,
5      pan VARCHAR(30) UNIQUE,
6      pin VARCHAR(4),
7      cvv VARCHAR(3),
8      expiring_date VARCHAR(20)
9  );
10

```

Below the editor, the 'Output' pane shows the execution results:

#	Time	Action	Message
1	14:34:19	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(20) PRIMARY KEY, iban VARCH...	0 row(s) affected

- Luego de crear la tabla cargamos los datos del archivo 'datos\_introducir\_credit'. Esto se debe realizar antes de referenciar en la tabla de hechos 'transaction' a la nueva tabla de dimensiones 'credit\_card'.

The screenshot shows a SQL IDE with a tab labeled 'datos\_introducir\_credit'. The SQL editor contains the following code:

```

1  -- Insertamos datos de credit_card
2
3  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2938', 'TR301950312213576817638661', '5424465566813633',
4  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2945', 'D026854763748537475216568689', '5142423821948828',
5  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2952', 'BG45IVQL52710525608255', '4556 453 55 5287', '459
6  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2959', 'CR7242477244335841535', '372461377349375', '3583'
7  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2966', 'BG72LKTQ15627628377363', '448566 886747 7265', '4
8  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2973', 'PT87806228135092429456346', '544 58654 54343 384'
9  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2980', 'DE39241881883086277136', '402400 7145845969', '56
10 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2987', 'GE89681434837748781813', '3763 747687 76666', '22

```

Below the editor, the 'Output' pane shows the execution results for 10 rows:

#	Time	Action	Message
265	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4779', 'FI91...	1 row(s) affected
266	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4786', 'SI51...	1 row(s) affected
267	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4793', 'HU9...	1 row(s) affected
268	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4800', 'SI97...	1 row(s) affected
269	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4807', 'LB1...	1 row(s) affected
270	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4814', 'MR4...	1 row(s) affected
271	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4821', 'LT2...	1 row(s) affected
272	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4828', 'BG1...	1 row(s) affected
273	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4835', 'PT3...	1 row(s) affected
274	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4842', 'SA2...	1 row(s) affected
275	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4849', 'SE2...	1 row(s) affected
276	14:36:18	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-4856', 'TR3...	1 row(s) affected

- Finalmente, para vincular las tablas, conectamos la nueva tabla de dimensiones creada, 'credit\_card', a la tabla de hechos 'transaction', mediante una FOREIGN KEY en 'transaction' que referencia a la PRIMARY KEY de 'credit\_card'.

- Notamos que la relación entre 'credit\_card' y 'company' se hace a través de la tabla de hechos, 'transaction', por lo que no es necesario hacer nada adicional para conectarlas.

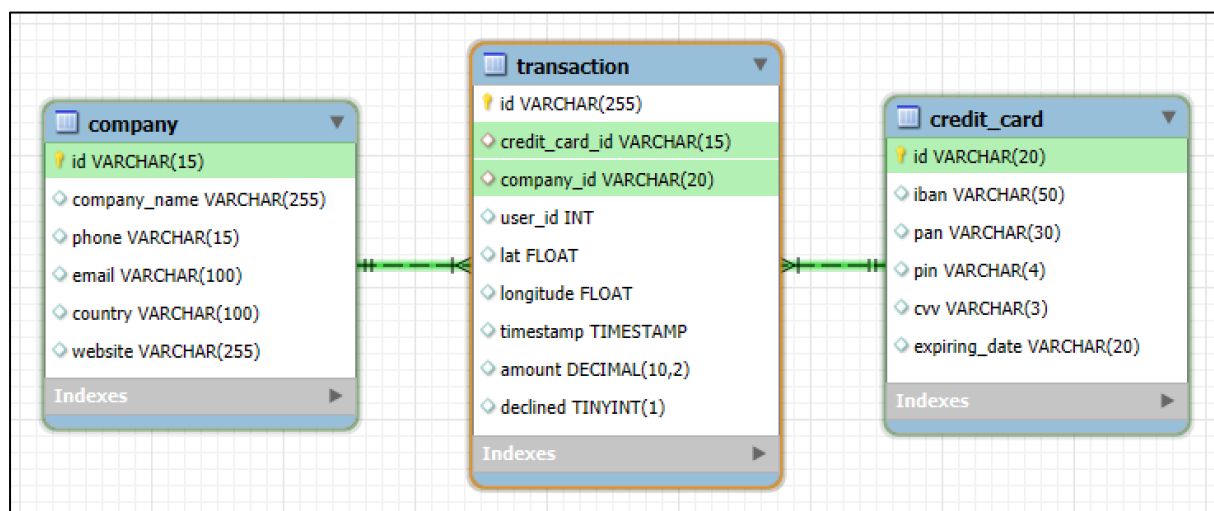
```

10
11 -- NIVEL 1 - EJERCICIO 1: Agregamos foreign key a tabla 'transaction'
12 -- referenciando a primary key de la tabla 'credit_card'
13 • ALTER TABLE transaction
14   ADD FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);
15
Output
Action Output
# Time Action Message
1 14:40:36 ALTER TABLE transaction ADD FOREIGN KEY (credit_card_id) REFERENCES credit_card(id) 587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0

```

- Observamos que la relación entre las tablas 'company' y 'transaction', así como la de 'credit\_card' con 'transaction', es 1:N, ya que una compañía puede tener muchas transacciones, pero una transacción corresponde a una sola compañía. Del mismo modo, una tarjeta de crédito puede realizar múltiples transacciones, pero cada transacción corresponde a una única tarjeta de crédito.

- Lo anterior se puede observar en el diagrama entidad-relación generado por MySQL:



## - Exercici 2

El departament de Recursos Humans ha identificat un error en el número de compte de l'usuari amb ID CcU-2938. La informació que ha de mostrar-se per a aquest registre és: R323456312213576817699999. Recorda mostrar que el canvi es va realitzar.

- Queremos realizar un cambio en el número de cuenta (IBAN) en la tabla 'credit\_card' para el usuario con 'id' igual a 'CcU-2938'.

- Primero, revisamos el valor actual del dato para ese usuario:

The screenshot shows a database IDE with a SQL editor and a result grid. The SQL editor contains the following code:

```
29
30  /* NIVEL 1 - EJERCICIO 2: Cambio de un registro - IBAN*/
31  -- Revisamos valor antes del cambio
32  • SELECT * FROM credit_card
33  WHERE id = 'CcU-2938';
```

The result grid displays the following data:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22
NULL	NULL	NULL	NULL	NULL	NULL

The Output pane shows the following message:

```
1 14:45:55 SELECT * FROM credit_card WHERE id = 'CcU-2938' LIMIT 0, 1000
1 row(s) returned
```

- Después, mediante UPDATE, editamos el valor del IBAN, especificando el usuario en la condición de la cláusula WHERE. Finalmente revisamos si el cambio se hizo correctamente, volviendo a extraer los datos para este usuario:

The screenshot shows a database IDE with a SQL editor and a result grid. The SQL editor contains the following code:

```
35  /* NIVEL 1 - EJERCICIO 2: Cambio de un registro - IBAN*/
36  -- realizamos el cambio
37  • UPDATE credit_card
38  SET iban = 'R323456312213576817699999'
39  WHERE id = 'CcU-2938';
40
41  -- Revisamos valor después del cambio
42  • SELECT * FROM credit_card
43  WHERE id = 'CcU-2938';
```

The result grid displays the following data:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	R323456312213576817699999	5424465566813633	3257	984	10/30/22
NULL	NULL	NULL	NULL	NULL	NULL

The Output pane shows the following messages:

```
1 14:47:00 UPDATE credit_card SET iban = 'R323456312213576817699999' WHERE id = 'CcU-2938'
1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

2 14:47:00 SELECT * FROM credit_card WHERE id = 'CcU-2938' LIMIT 0, 1000
1 row(s) returned
```

### - Exercici 3

En la taula "transaction" ingressa un nou usuari amb la següent informació:

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lat	829.999
longitude	-117.999
amount	111.11
declined	0

- Al intentar ingresar un nuevo usuario con la información entregada, MySQL nos entrega un error, ya que estamos ingresando valores en variables que funcionan como FOREIGN KEY, que no están presentes como PRIMARY KEY en las tablas de dimensiones ('credit\_card' y 'company')

```
37 SET iban = 'R323456312213576817699999'
38 WHERE id = 'CcU-2938';
39
40 -- Revisamos valor después del cambio
41 • SELECT * FROM credit_card
42 WHERE id = 'CcU-2938';
43
44 /* NIVEL 1 - EJERCICIO 3: Insertar datos nuevos a tabla ya creada - transaction*/
45 -- Insertamos datos de transaction
46 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
47 VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0');
48
49
```

Output

#	Time	Action	Message
1	10:57:43	UPDATE credit_card SET iban = 'R323456312213576817699999' WHERE id = 'CcU-2938'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
2	10:57:43	SELECT * FROM credit_card WHERE id = 'CcU-2938' LIMIT 0, 1000	1 row(s) returned
3	11:01:07	SELECT * FROM credit_card WHERE id = 'CcU-9999' LIMIT 0, 1000	0 row(s) returned
4	11:11:48	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, decli...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (transactions'

- Una solución, es crear nuevos registros en las tablas de dimensiones referenciadas en la tabla 'transaction', que contengan los valores que estamos ingresando en sus FOREIGN KEY. Decidimos ingresar solamente los valores que se nos han entregado, dejando NULL los otros campos, dejando claro que se espera que el departamento correspondiente entregue los valores para completarla.

- Por ahora, lo anterior se debe realizar solamente en las tablas 'credit\_card' y 'company', pero más adelante debemos realizarlo también para la tabla de dimensiones 'user', luego de que esta sea creada.

tasca3\* x datos\_introducir\_credit estructura\_datos\_user datos\_introducir\_user (1) estructura\_dades dades\_introduir

Limit to 1000 rows

```

50
51 • INSERT INTO company (id)
52   VALUES ('b-9999');
53 • SELECT *
54   FROM company
55   WHERE id = 'b-9999';

```

**Result Grid** | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: [IA](#)

	id	company_name	phone	email	country	website
▶	b-9999	NULL	NULL	NULL	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

**Output**

Action Output

#	Time	Action	Message
✓ 1	09:27:46	INSERT INTO company (id) VALUES (b-9999)	1 row(s) affected
✓ 2	09:27:46	SELECT * FROM company WHERE id = 'b-9999' LIMIT 0, 1000	1 row(s) returned

tasca3\* x datos\_introducir\_credit estructura\_datos\_user datos\_introducir\_user (1) estructura\_dades dades\_introduir

Limit to 1000 rows

```

57 • INSERT INTO credit_card (id)
58   VALUES ('CcU-9999');
59 • SELECT *
60   FROM credit_card
61   WHERE id = 'CcU-9999';

```

**Result Grid** | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: [IA](#)

	id	iban	pan	pin	cvv	expiring_date
▶	CcU-9999	NULL	NULL	NULL	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

**Output**

Action Output

#	Time	Action	Message
✓ 1	09:31:30	INSERT INTO credit_card (id) VALUES ('CcU-9999')	1 row(s) affected
✓ 2	09:31:30	SELECT * FROM credit_card WHERE id = 'CcU-9999' LIMIT 0, 1000	1 row(s) returned

tasca3\* x datos\_introducir\_credit estructura\_datos\_user datos\_introducir\_user (1) estructura\_dades dades\_introduir

Limit to 1000 rows

```

62
63 /*Volvemos a intentar el ingreso solicitado a la tabla 'transaction'*/
64 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
65   VALUES ('10881D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0');
66
67

```

**Output**

Action Output

#	Time	Action	Message
✓ 1	09:32:48	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) ...	1 row(s) affected



## - Exercici 4

Des de recursos humans et sol·liciten eliminar la columna "pan" de la taula credit\_card. Recorda mostrar el canvi realitzat.

- Para eliminar una columna de una tabla ya existente, utilizamos ALTER TABLE y DROP COLUMN. A continuación, vemos la ejecución del código y revisión del resultado:

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains the following code:

```
66
67 /* NIVEL 1 - EJERCICIO 4: Eliminar columna de tabla*/
68 -- Eliminamos columna 'pan' de tabla 'credit_card'
69 • ALTER TABLE credit_card
70   DROP COLUMN pan;
71 -- Revisamos si la columna fue eliminada
72 • SELECT *
73   FROM credit_card;
```

The results pane shows a table with 5 columns: id, iban, pin, cvv, and expiring\_date. The table contains 6 rows of data:

id	iban	pin	cvv	expiring_date
CcU-2938	R323456312213576817699999	3257	984	10/30/22
CcU-2945	DO26854763748537475216568689	9080	887	08/24/23
CcU-2952	BG45IVQL52710525608255	4598	438	06/29/21
CcU-2959	CR7242477244335841535	3583	667	02/24/23
CcU-2966	BG72LKTQ15627628377363	4900	130	10/29/24

The output pane shows the execution of the SQL statements:

#	Time	Action	Message
1	09:35:06	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	09:35:06	SELECT * FROM credit_card LIMIT 0, 1000	276 row(s) returned

- Notamos que el cambio afecta a 276 registros, porque tenemos un registro adicional creado en el ejercicio 3.

## Nivell 2

### Exercici 1

Elimina de la taula transaction el registre amb ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de dades.

- Eliminamos un registro de una tabla con DELETE FROM, especificando en la cláusula WHERE que solamente queremos borrar el registro de la 'id' entregada.
- Luego verificamos si efectivamente se ha borrado el registro correspondiente.

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains the following code:

```
75 #-----#
76 /* NIVEL 2 - EJERCICIO 1: Eliminar registro (fila) de una tabla*/
77 -- DELETE FROM table_name WHERE condition;
78 -- Eliminamos registro
79 • DELETE FROM transaction
80 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
81 -- Revisamos si el registro fue eliminado
82 • SELECT *
83 FROM transaction
84 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
85
```

The results pane shows a table with the following columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined. The table is currently empty, with all cells showing NULL.

Below the table, the 'Output' section shows the 'Action Output' for the executed SQL statements:

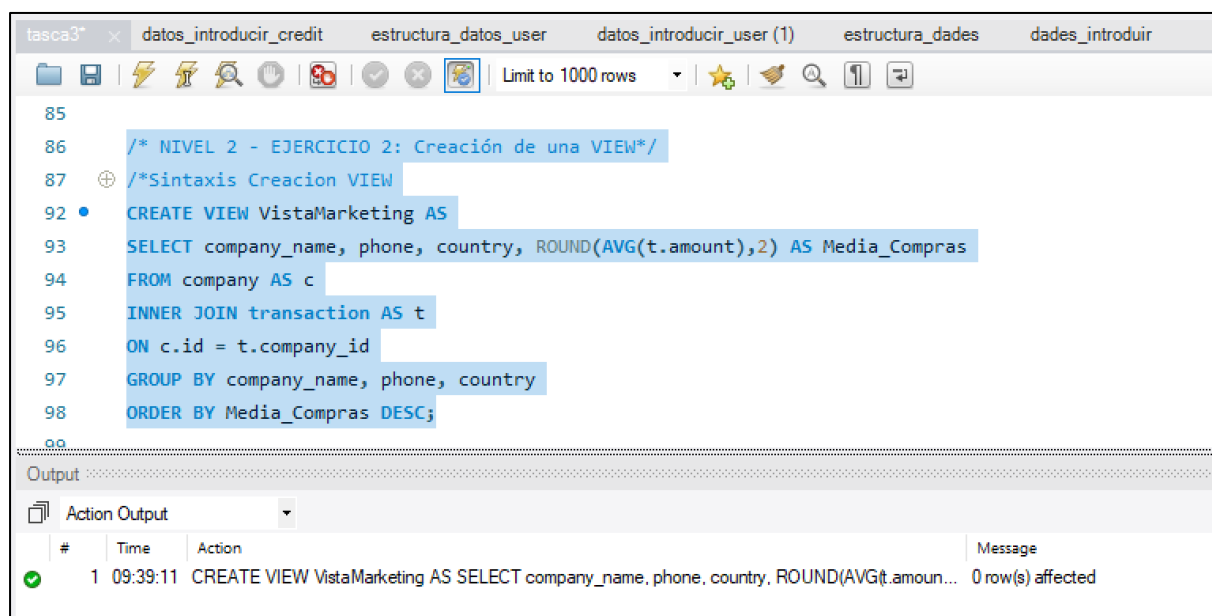
#	Time	Action	Message
1	09:37:20	DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02'	1 row(s) affected
2	09:37:20	SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02' LIMIT 0, ...	0 row(s) returned



## Exercici 2

La secció de màrqueting desitja tenir accés a informació específica per a realitzar anàlisi i estratègies efectives. S'ha sol·licitat crear una vista que proporcioni detalls clau sobre les companyies i les seves transaccions. Serà necessària que creïs una vista anomenada VistaMarketing que contingui la següent informació: Nom de la companyia. Telèfon de contacte. País de residència. Mitjana de compra realitzat per cada companyia. Presenta la vista creada, ordenant les dades de major a menor mitjana de compra.

- Creamos la vista solicitada usando CREATE VIEW.
- Unimos las tablas 'company' y 'transaction' mediante INNER JOIN, seleccionamos los datos solicitados, incluyendo la variable calculada 'Media\_Compras'.
- Agregamos con GROUP BY por todas las variables no agregadas en la cláusula SELECT, y ordenamos con ORDER BY por la 'Media\_Compras' de forma descendente. Notar que obtenemos 101 registros por el registro creado en Ejercicio 3 – Nivel 1-



```
85
86  /* NIVEL 2 - EJERCICIO 2: Creación de una VIEW*/
87  /*Sintaxis Creacion VIEW
92  • CREATE VIEW VistaMarketing AS
93  SELECT company_name, phone, country, ROUND(AVG(t.amount),2) AS Media_Compras
94  FROM company AS c
95  INNER JOIN transaction AS t
96  ON c.id = t.company_id
97  GROUP BY company_name, phone, country
98  ORDER BY Media_Compras DESC;
99
```

Output

#	Time	Action	Message
✓ 1	09:39:11	CREATE VIEW VistaMarketing AS SELECT company_name, phone, country, ROUND(AVG(t.amoun...	0 row(s) affected

- Luego seleccionamos todo el contenido de la VIEW para revisarlo:

The screenshot shows a database management interface with several tabs at the top: 'tasca3\*', 'datos\_introducir\_credit', 'estructura\_datos\_user', 'datos\_introducir\_user (1)', 'estructura\_dades', and 'dades\_introduir'. The active tab is 'datos\_introducir\_credit'. The SQL editor shows the following code:

```

99
100  -- Revision de la VIEW
101  •  SELECT *
102     FROM VistaMarketing;
103

```

Below the editor, the 'Result Grid' is displayed with the following data:

	company_name	phone	country	Media_Compras
▶	Eget Ipsum Ltd	03 67 44 56 72	United States	473.08
	Non Magna LLC	06 71 73 13 17	United Kingdom	468.35
	Sed Id Limited	07 28 18 18 13	United States	461.21
	Justo Eu Arcu Ltd	08 42 56 71 52	Italy	443.64
	Eget Tincidunt Dui Institute	05 35 93 32 44	Netherlands	442.52
	Viverra Donec Foundation	03 33 12 32 73	United Kingdom	442.28
	Vestibulum Lorem PC	02 02 87 33 40	Belgium	434.06

Below the result grid, there is a section for 'Output' and 'Action Output'. The 'Action Output' table shows the following entry:

#	Time	Action	Message
✓ 1	09:40:59	SELECT * FROM VistaMarketing LIMIT 0, 1000	101 row(s) returned

### Exercici 3

Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany"

- Seleccionamos los datos de la vista con la condición solicitada, de la misma manera que seleccionamos datos de una tabla:

The screenshot shows a database management interface with a SQL editor and a results grid. The SQL query is as follows:

```
104
105 /* NIVEL 2 - EJERCICIO 3: Selección de datos de una VIEW*/
106 • SELECT *
107 FROM VistaMarketing
108 WHERE country = 'Germany'
109 ORDER BY Media_Compras DESC;
110
```

The results grid displays the following data:

company_name	phone	country	Media_Compras
Aliquam PC	01 45 73 52 16	Germany	385.27
Ac Industries	09 34 65 40 60	Germany	289.65
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90
Nunc Interdum Incorporated	05 18 15 48 13	Germany	244.03
Augue Foundation	06 88 43 15 63	Germany	240.80
Ac Fermentum Incorporated	06 85 56 52 33	Germany	206.47
Auctor Mauris Corp.	05 62 87 14 41	Germany	184.31
Convallis In Incorporated	06 66 57 29 50	Germany	156.73

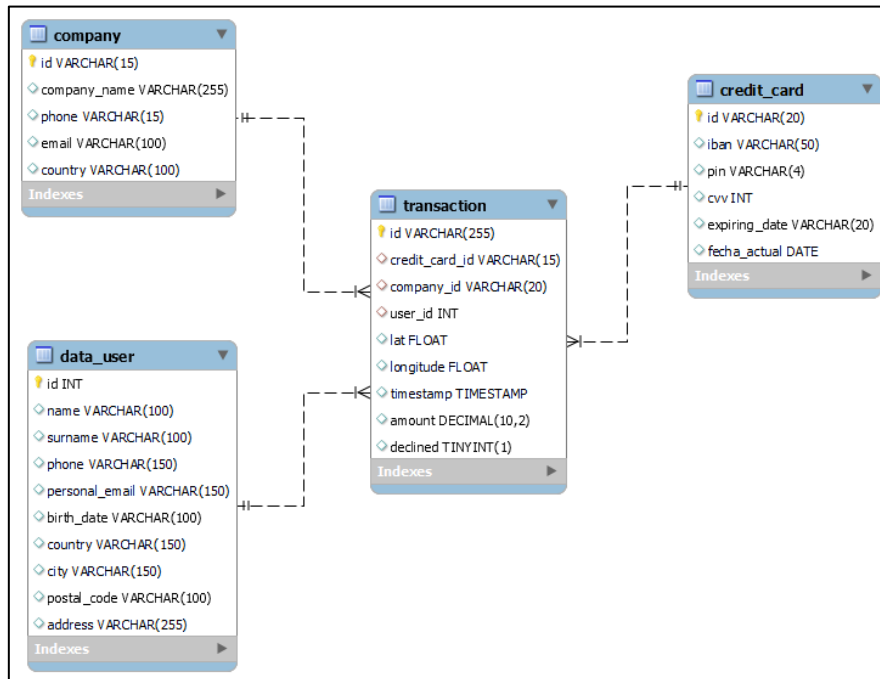
The output section shows the execution of the query:

#	Time	Action	Message
1	09:44:00	SELECT * FROM VistaMarketing WHERE country = 'Germany' ORDER BY Media_Compras DESC ...	8 row(s) returned

## Nivell 3

### Exercici 1

La setmana vinent tindràs una nova reunió amb els gerents de màrqueting. Un company del teu equip va realitzar modificacions en la base de dades, però no recorda com les va realitzar. Et demana que l'ajudis a deixar els comandos executats per a obtenir el següent diagrama:

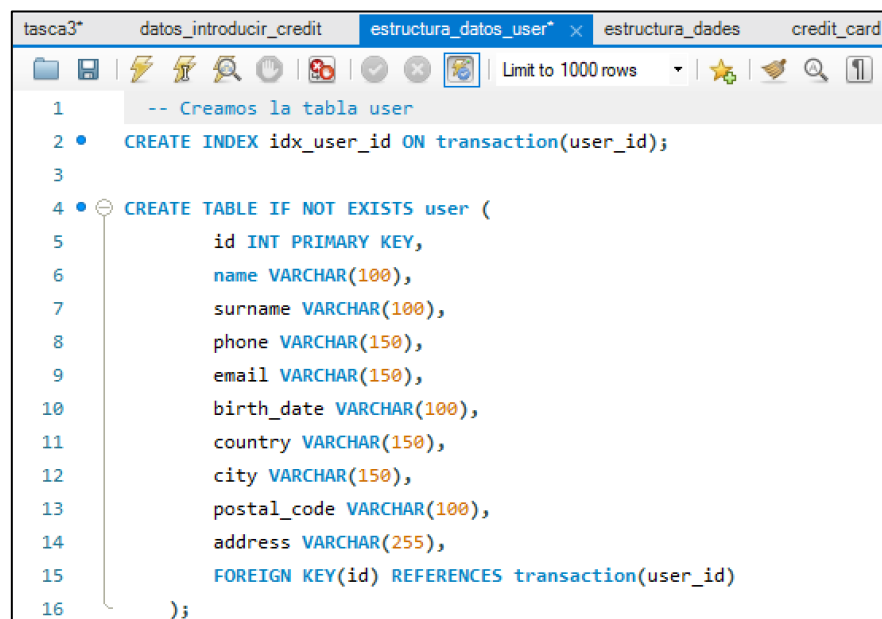


## Recordatori

En aquesta activitat, és necessari que descriguis el "pas a pas" de les tasques realitzades. És important realitzar descripcions senzilles, simples i fàcils de comprendre. Per a realitzar aquesta activitat hauràs de treballar amb els arxius denominats "estructura\_dades\_user" i "dades\_introduir\_user"

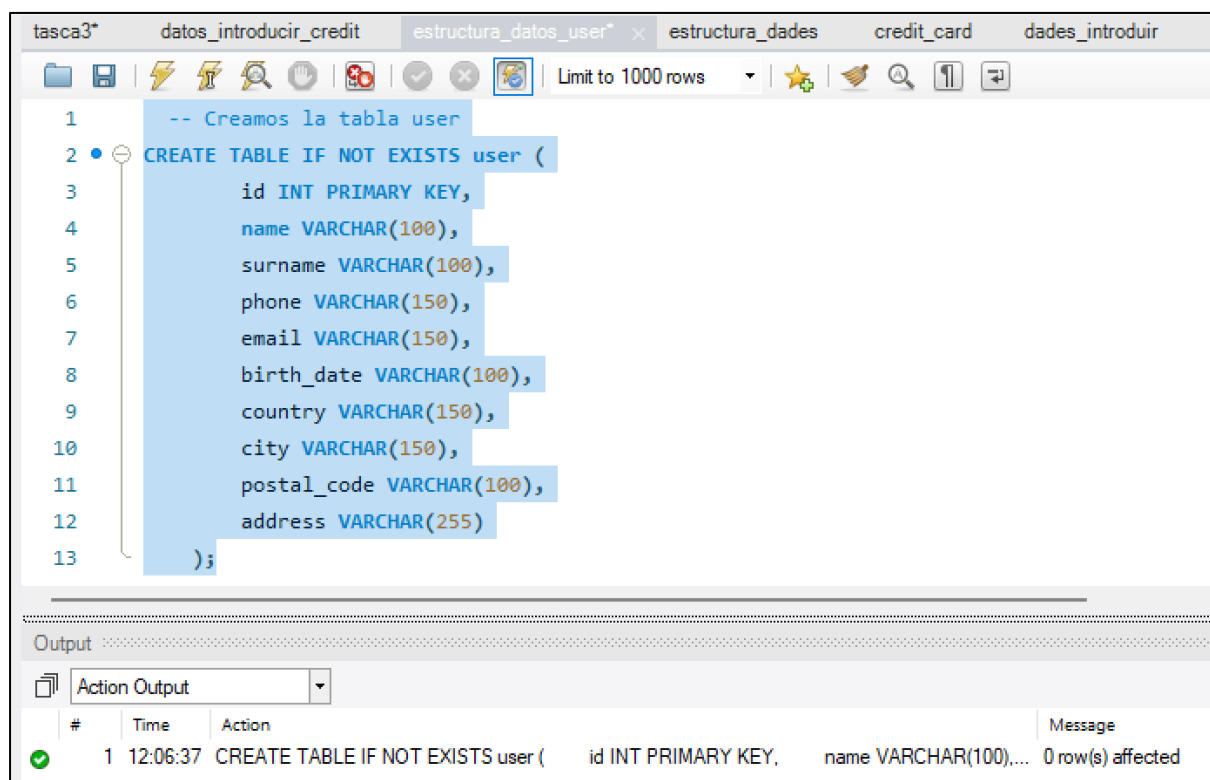
- Primero, debemos crear la tabla usuario. Revisamos el archivo 'estructura\_datos\_user' para asegurarnos de que esté bien.
- Notamos que las variables, sus nombre y tipos de datos, están todas bien. El único error es la inclusión de una foreign key, que no corresponde en una tabla de dimensiones como 'user'. Eliminamos la foreign key junto al índice creado para ella y ejecutamos el código:

Código antes de la corrección:



```
1  -- Creamos la tabla user
2  • CREATE INDEX idx_user_id ON transaction(user_id);
3
4  • CREATE TABLE IF NOT EXISTS user (
5      id INT PRIMARY KEY,
6      name VARCHAR(100),
7      surname VARCHAR(100),
8      phone VARCHAR(150),
9      email VARCHAR(150),
10     birth_date VARCHAR(100),
11     country VARCHAR(150),
12     city VARCHAR(150),
13     postal_code VARCHAR(100),
14     address VARCHAR(255),
15     FOREIGN KEY(id) REFERENCES transaction(user_id)
16 );
```

Código después de la corrección:



```
1  -- Creamos la tabla user
2  • CREATE TABLE IF NOT EXISTS user (
3      id INT PRIMARY KEY,
4      name VARCHAR(100),
5      surname VARCHAR(100),
6      phone VARCHAR(150),
7      email VARCHAR(150),
8      birth_date VARCHAR(100),
9      country VARCHAR(150),
10     city VARCHAR(150),
11     postal_code VARCHAR(100),
12     address VARCHAR(255)
13 );
```

Output

Action Output

#	Time	Action	Message
✓ 1	12:06:37	CREATE TABLE IF NOT EXISTS user ( id INT PRIMARY KEY, name VARCHAR(100)...	0 row(s) affected

- El siguiente paso es poblar la tabla con los datos de 'datos\_introducir\_user (1)'. Esto se debe hacer antes de conectar la tabla 'user' a la tabla de hechos 'transaction', mediante una foreign key en 'transaction'. Obviamos la primera y última línea del código que hacen referencia a la foreign key que eliminamos.

The screenshot shows a SQL script in a window titled 'datos\_introducir\_user (1)'. The script contains 11 INSERT statements for the 'user' table, each with a unique ID and specific user details. The statements are numbered 268 through 278. The final statement on line 280 is 'SET foreign\_key\_checks = 1;'. Below the script, the 'Output' pane shows the execution results for lines 268 to 276, all of which were successful.

```

268 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "265", "Chloe", "Keith", "021
269 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "266", "Aiko", "Chaney", "028
270 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "267", "Ocean", "Nelson", "07
271 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "268", "Clark", "Olson", "029
272 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "269", "Haley", "Fitzpatrick"
273 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "270", "Elton", "Roberson", "
274 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "271", "Leandra", "Cherry", "
275 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "272", "Hedwig", "Gilbert", "
276 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "273", "Hilary", "Ferguson",
277 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "274", "Jameson", "Hunt", "0
278 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "275", "Kenyon", "Hartman", "
279
280 • SET foreign_key_checks = 1;

```

#	Time	Action	Message	Duration
✓ 268	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 269	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 270	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 271	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 272	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.015 se
✓ 273	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 274	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 275	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se
✓ 276	12:19:02	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, ad...	1 row(s) affected	0.000 se

- A continuación, agregamos un registro en 'user' para que sea consistente con el ingreso del registro en 'transaction', del Ejercicio 3 – Nivel 1.

The screenshot shows a database IDE with the following SQL code in the editor:

```

110 #-----#
111 /* NIVEL 3 - EJERCICIO 1: Ingreso registro en user para conectar con registro conectado en EJERCICIO 3 - NIVEL 1*/
112 • INSERT INTO user (id)
113   VALUES ('9999');
114 • SELECT *
115   FROM user
116   WHERE id = '9999';
117
118
119
120

```

Below the editor, the 'Result Grid' shows the result of the SELECT query:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
9999	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The 'Output' pane shows the 'Action Output' for the executed queries:

#	Time	Action	Message
1	10:20:34	INSERT INTO user (id) VALUES ('9999')	1 row(s) affected
2	10:20:34	SELECT * FROM user WHERE id = '9999' LIMIT 0, 1000	1 row(s) returned

- Luego creamos una PRIMARY KEY en 'transaction' para conectar la tabla 'user' al resto de las tablas:

The screenshot shows a database IDE with the following SQL code in the editor:

```

117
118 /*Creacion de una Foreign Key transaction->user*/
119 • ALTER TABLE transaction
120   ADD FOREIGN KEY (user_id) REFERENCES user(id);
121

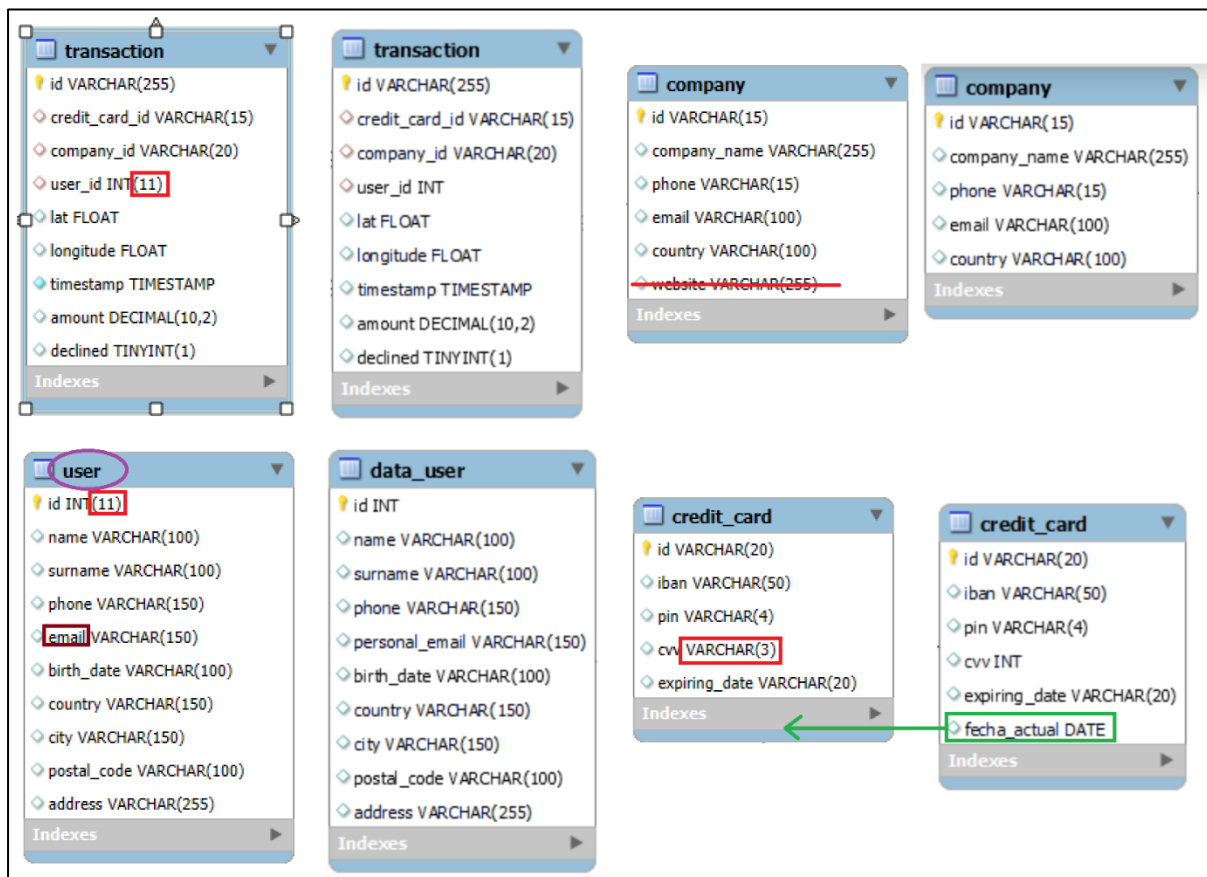
```

The 'Output' pane shows the 'Action Output' for the executed query:

#	Time	Action	Message
1	10:21:40	ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES user(id)	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0



- Ahora comparamos las tablas actuales con las deseadas en detalle:



- **Cambiar tipos de datos:**

Cambiamos el tipo de datos de 'transaction.user\_id', 'user.id' y 'credit\_card.cvv'.

- **Cambiar nombre campo:**

Cambiamos nombre de campo 'user.email' a 'user.personal\_email'.

- **Cambiar nombre tabla:**

Cambiamos nombre de tabla 'user' a 'data\_user'.

- **Eliminar campo:**

Eliminamos el campo 'company.website'

- **Agregar campo:**

Agregamos el campo 'credit\_card.fecha\_actual', con datos del tipo DATE.

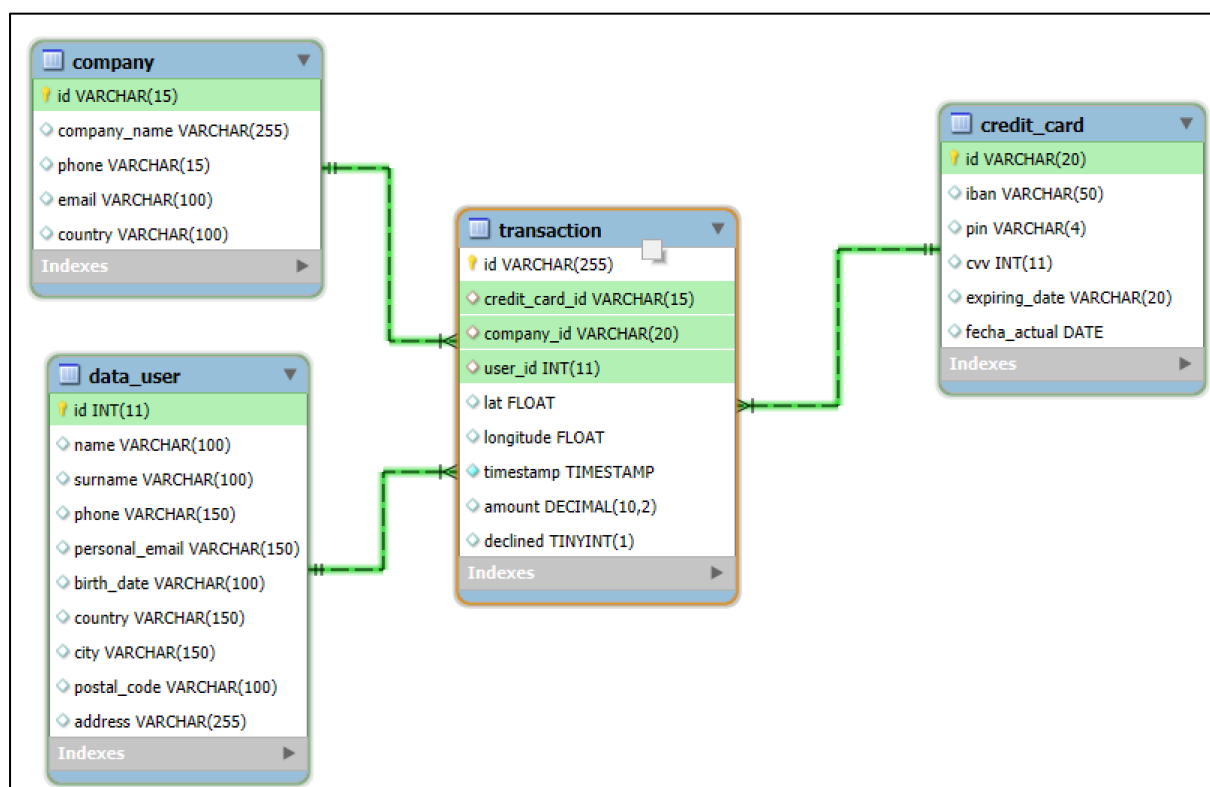
```

122  /*NIVEL 3 - EJERCICIO 1: AJUSTES DE TABLAS*/
123  -- Cambiar de tipos de datos:
124  • ALTER TABLE transaction MODIFY COLUMN user_id INT;
125  • ALTER TABLE user MODIFY COLUMN id INT;
126  • ALTER TABLE credit_card MODIFY COLUMN cvv INT;
127
128  -- Cambiar nombre campo:
129  • ALTER TABLE user CHANGE email personal_email VARCHAR(150);
130
131  -- Cambiar nombre tabla 'user' a 'data_user'
132  • ALTER TABLE user RENAME TO data_user;
133
134  -- Eliminar campo 'website' de tabla 'company'
135  • ALTER TABLE company DROP COLUMN website;
136
137  -- Agregar campo 'fecha_actual' a tabla 'credit_card'
138  • ALTER TABLE credit_card ADD fecha_actual DATE;

```

#	Time	Action	Message
✓ 1	12:08:08	ALTER TABLE transaction MODIFY COLUMN user_id INT	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 2	12:08:11	ALTER TABLE user MODIFY COLUMN id INT	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 3	12:08:15	ALTER TABLE credit_card MODIFY COLUMN cvv INT	277 row(s) affected Records: 277 Duplicates: 0 Warnings: 0
✓ 4	12:08:20	ALTER TABLE user CHANGE email personal_email VARCHAR(150)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 5	12:08:27	ALTER TABLE user RENAME TO data_user	0 row(s) affected
✓ 6	12:08:34	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 7	12:08:37	ALTER TABLE credit card ADD fecha actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Después de todo lo anterior, el diagrama relacional nos queda así:



## Exercici 2

L'empresa també et sol·licita crear una vista anomenada "InformeTecnico" que contingui la següent informació:

- ID de la transacció
- Nom de l'usuari/ària
- Cognom de l'usuari/ària
- IBAN de la targeta de crèdit usada.
- Nom de la companyia de la transacció realitzada.

Assegura't d'incloure informació rellevant de totes dues taules i utilitza àlies per a canviar de nom columnes segons sigui necessari.

Mostra els resultats de la vista, ordena els resultats de manera descendent en funció de la variable ID de transaction.

Para tener claridad de los datos solicitados, y de en que tablas se encuentra, creamos la siguiente tabla:

Dato Solicitado	Tabla donde se encuentra	Nombre de Variable
ID de la transacció	transaction	id
Nom de l'usuari/ària	data_user	name
Cognom de l'usuari/ària	data_user	surname
IBAN de la targeta de crèdit usada	credit_card	id
Nom de la companyia de la transacció realitzada	company	company_name

- Conectando todas las tablas mediante JOINS, creamos la vista solicitada.

```
140 /* NIVEL 3 - EJERCICIO 2: Seleccion de datos con de diversas tablas*/
141 /*Syntaxis multiples JOINS. 3 tablas de Dimensiones y 1 tabla de Hechos
153 • CREATE VIEW InformeTecnico AS
154 SELECT
155     t.id AS id_transaccion,
156     data_user.name AS nombre,
157     data_user.surname AS apellido,
158     credit_card.id AS IBAN,
159     company.company_name AS nombre_compañia
160 FROM transaction AS t
161 INNER JOIN company ON t.company_id = company.id
162 INNER JOIN credit_card ON t.credit_card_id = credit_card.id
163 INNER JOIN data_user ON t.user_id = data_user.id
164 ORDER BY id_transaccion DESC;
165
166 • SELECT *
167 FROM InformeTecnico;
```

Output

#	Time	Action	Message
1	12:35:12	CREATE VIEW InformeTecnico AS SELECT t.id AS id_transaccion, data_user.name AS nombre, da...	0 row(s) affected

- Finalmente seleccionamos todos los datos de la vista creada para revisar que todo esté en orden:

The screenshot shows a database management interface with a query editor and a results grid. The query editor contains the following SQL statement:

```
SELECT *
FROM InformeTecnico;
```

The results grid displays the following data:

id_transaccion	nombre	apellido	IBAN	nombre_compañia
FE96CE47-8D59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	CcU-2945	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E4646B	Molly	Gilliam	CcU-4849	Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	CcU-4331	Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	CcU-3960	Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	CcU-3232	Neque Tellus Imperdiet Corp.
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	Hakeem	Alford	CcU-4219	Nunc Interdum Incorporated
FBD7E0D6-BA6B-F5BC-0CA9-EA4B8760100C	Hedwig	Gilbert	CcU-3281	Mauris Id Inc.
FAC76A80-8448-69AA-E892-426C2F12621C	Slade	Poole	CcU-4576	Arcu LLP
FAAD3FFC-1A17-E141-43D3-359A5BA7CB3B	Hedwig	Gilbert	CcU-3183	Lorem Eu Incorporated
FA053936-75D8-85FA-490D-9B624E1B920A	Hedwig	Gilbert	CcU-3197	Non Justo Corp.
F85A7D75-2778-9D75-D776-3F41A828DE88	Sarah	Beck	CcU-3393	Ut Semper Foundation
F843DC08-CCB5-2444-1B4E-5966289FBA8B	Jasper	Landry	CcU-3393	Ut Semper Foundation

The output section shows the following message:

```
1 12:36:42 SELECT * FROM InformeTecnico LIMIT 0, 1000
587 row(s) returned
```