

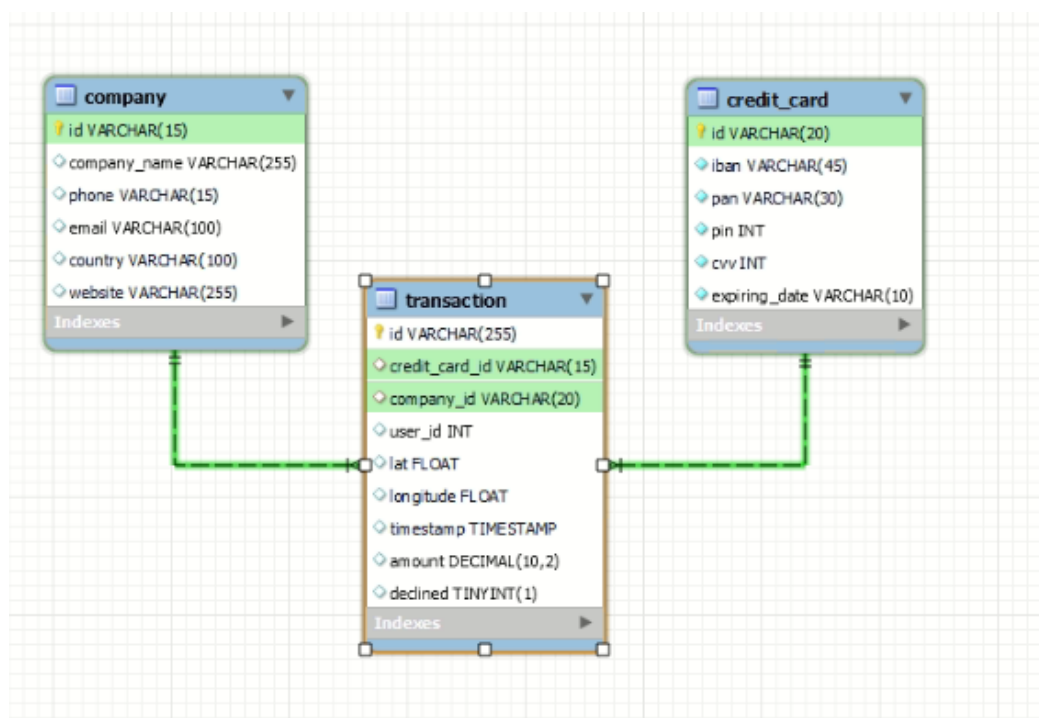
### SPRINT 3 - NIVEL 1 - EJERCICIO 1

Se pide la creación de la estructura de la tabla para posteriormente incluir en la misma la data, para finalmente, añadir tanto índices como las Entidades-Relación.

Con el comando CREATE TABLE creamos la estructura adecuada para albergar la data proporcionada por el ejercicio. Con INSERT VALUE añadimos la data a la estructura creada

Siendo que la tabla TRANSACTIONS es la tabla principal creamos la relación 1 a N, en la que las foreign key son company\_id a id para las empresas, y creditcard\_id a id para las tarjetas de crédito.

```
Sprint_1_FULL Sprint_2_FULL alter_table SQL File 7* alumno salon SPRINT_3_PDF_FULL SQL File 9* transaction-Table SPRINT_3_PDF_FULL
1 # Creamos la Estructura de la tabla para las Tarjetas de Crédito
2 CREATE TABLE `transactions`.`credit_card` (
3   `id` VARCHAR(20) NOT NULL,
4   `iban` VARCHAR(45) NOT NULL,
5   `pan` VARCHAR(30) NOT NULL,
6   `pin` INT NOT NULL,
7   `cvv` INT NOT NULL,
8   `expiring_date` VARCHAR(10) NOT NULL,
9   PRIMARY KEY (`id`)
10 );
11 # Introducimos VALORES descargados INTO la tabla creditcard
12 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2938', 'TR381950312213576817638661', '5424465566813633', '3257', '904', '10/30/2019');
13 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2945', 'D026854763748537475216568689', '5142423821948828', '9080', '887', '08/24/2019');
14 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2952', 'B645IVQL52710525608255', '4556 453 55 5287', '4598', '438', '06/29/21');
15 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2959', 'CR7242477244335841535', '372461377349375', '3583', '667', '02/24/23');
16 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2966', 'B672LKTQ15627628377363', '448566 886747 7265', '4900', '130', '10/29/24');
17 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2973', 'PT87806228135092429456346', '544 58654 54343 384', '8760', '887', '01/30/2019');
18 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2980', 'DE39241881883086277136', '402400 7145845969', '5075', '596', '07/24/22');
19 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2987', 'GE89681434837748781813', '3763 747687 76666', '2298', '797', '10/31/22');
20 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2994', 'BH62714428368066765294', '344283273252593', '7545', '595', '02/28/22');
21 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3001', 'CV49087426654774581266832110', '511722 924833 2244', '9562', '867', '09/01/2019');
22 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3008', 'LU507216693616119230', '4485744464433884', '1856', '740', '04/05/25');
23 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3015', 'PS119398216295715968342456821', '3784 662233 17389', '3246', '822', '01/01/2019');
24 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3022', 'GT91695162850556977423121857', '5164 1379 4842 3951', '5610', '342', '04/01/2019');
25 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3029', 'AZ62317413982441418123739746', '3429 279566 77631', '9708', '505', '09/01/2019');
26 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3036', 'AZ39336002925842865843941994', '3768 451556 48766', '2232', '565', '10/10/2019');
27 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3043', 'TN6488143310514852179535', '455676 6437463635', '5969', '196', '06/07/25');
28 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3050', 'FR5167744369175836831854477', '4024007123722', '4834', '126', '10/09/23');
29 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3057', 'LU931822574697545215', '3484 621767 21237', '6805', '848', '09/14/25');
30 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3064', 'PS146905545449253372627273133', '3467 732741 26810', '3865', '498', '06/01/2019');
```



## SPRINT 3 - NIVEL 1 - EJERCICIO 2

Se pide el UPDATE del iban de una empresa con un id determinado ccu2938

Por ello hago una consulta en la tabla `credit\_card` para seleccionar todas las columnas (`\*`) donde el valor de la columna `id` es igual a `CcU-2938`. En otras palabras, busca y devuelve todas las filas de la tabla `credit\_card` que tienen el valor específico `CcU-2938` en la columna `id`.

```
# SPRINT 3 - NIVEL 1 - EJERCICIO 2

• SELECT *
  FROM credit_card
  WHERE id = 'CcU-2938';

# sustituir 'TR301950312213576817638661' por ' R323456312213576817699999 ' en campos indicados

• UPDATE credit_card SET iban='R323456312213576817699999'
  WHERE id = 'CcU-2938';
```

## Output

306

Result Grid						
Filter Rows:						
Edit: Export/Import: Wrap Cell Content:						
	id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	R323456312213576817699999	5424465566813633	3257	984	10/30/22
*	NULL	NULL	NULL	NULL	NULL	NULL

### SPRINT 3 - NIVEL 1 - EJERCICIO 3

Añadir un registro nuevo a la tabla transactions, siendo necesario desactivar temporalmente la llave foránea por estar configurada en modo cascade.

El código temporalmente la comprobación de llaves foráneas para permitir la inserción de datos que podrían violar las restricciones de integridad referencial.

Inserta un nuevo registro en la tabla 'transaction'.

Vuelve a activar la comprobación de llaves foráneas para restaurar las restricciones de integridad referencial.

Selecciona todos los datos de la tabla 'transaction' donde el ID es igual a '108B1D1D-5B23-A76C-55EF-C568E49A99DD'.

CÓDIGO:

```
# SPRINT 3 - NIVEL 1 - EJERCICIO 3
# añadir un registro a la tabla de transacciones, siendo necesario desactivar la llave foránea de forma temporal por estan en cascade
# Entrada de un nuevo registro (desactivamos temporalmente la llave foránea)
-- Desactivación temporal de la comprobación de llaves foráneas
SET FOREIGN_KEY_CHECKS = 0; -- Esto se hace para evitar que las restricciones de clave foránea se apliquen temporalmente.

-- Inserción de datos en la tabla 'transaction'
INSERT INTO transaction (id, credit_card_id, user_id, lat, longitude, amount, declined)
VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CCU-9999', '9999', '829.999', '-117.999', 111.11, '0');

-- Activación de la comprobación de llaves foráneas
SET FOREIGN_KEY_CHECKS = 1; -- Esto vuelve a habilitar la verificación de las restricciones de clave foránea.

-- Consulta para seleccionar todos los datos de la tabla 'transaction' donde el ID es igual a '108B1D1D-5B23-A76C-55EF-C568E49A99DD'
SELECT * FROM transaction
WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
```

OUTPUT:

[illegible]

### **SPRINT 3 - NIVEL 1 - EJERCICIO 4**

Eliminar la columna pan de la tabla credit\_card

Esta rutina Utiliza la declaración ALTER TABLE para modificar la estructura de la tabla credit\_card.

Con la instrucción DROP COLUMN, elimina la columna llamada pan de la tabla credit\_card.

Después de ejecutar la eliminación de la columna, se utiliza la instrucción DESCRIBE para mostrar la estructura actualizada de la tabla credit\_card. Esto proporciona información sobre las columnas restantes en la tabla después de la eliminación de la columna pan.

CODIGO:

```
-- SPRINT 3 - NIVEL 1 - EJERCICIO 4
-- Eliminar la columna (drop) de la tabla tarjetas de crédito

-- Eliminar la columna 'pan' de la tabla 'credit_card'
• ALTER TABLE credit_card
  DROP COLUMN pan;

-- Describir la estructura actualizada de la tabla 'credit_card'
• DESCRIBE credit_card;
```

OUTPUT:

Result Grid   Filter Rows:   Export:   Wrap Cell Content:						
	Field	Type	Null	Key	Default	Extra
▶	id	varchar(20)	NO	PRI	NULL	
	iban	varchar(45)	NO		NULL	
	pin	int	NO		NULL	
	cvv	int	NO		NULL	
	expiring_date	varchar(10)	NO		NULL	

### SPRINT 3 - NIVEL 2 - EJERCICIO 1

Eliminar un registro determinado de la tabla credit\_card

Usamos delete previo desactivación de la tabla foreign key '0'0' para crear un código que desactiva temporalmente la comprobación de llaves foráneas para permitir la eliminación del registro sin verificar las restricciones de integridad referencial.

Utiliza la instrucción DELETE para eliminar el registro de la tabla credit\_card donde el ID es '02C6201E-D90A-1859-B4EE-88D2986D3B02'.

Vuelve a activar la comprobación de llaves foráneas para restaurar las restricciones de integridad referencial.

Realiza una consulta para seleccionar y mostrar todos los datos de la tabla credit\_card donde el ID es '02C6201E-D90A-1859-B4EE-88D2986D3B02'. Esto se hace para confirmar que el registro ha sido eliminado correctamente.

CÓDIGO:

```
-- SPRINT 3 - NIVEL 2 - EJERCICIO 1
-- Eliminar un registro concreto de la tabla credit_card

-- Desactivación temporal de la comprobación de llaves foráneas
• SET FOREIGN_KEY_CHECKS = 0; -- Esto se hace para evitar que las restricciones de clave foránea se apliquen temporalmente.

-- Elimina el registro de la tabla 'credit_card' donde el ID es '02C6201E-D90A-1859-B4EE-88D2986D3B02'
• DELETE FROM credit_card
  WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';

-- Activación de la comprobación de llaves foráneas
• SET FOREIGN_KEY_CHECKS = 1; -- Esto vuelve a habilitar la verificación de las restricciones de clave foránea.

-- Consulta para seleccionar todos los datos de la tabla 'credit_card' donde el ID es '02C6201E-D90A-1859-B4EE-88D2986D3B02'
• SELECT * FROM credit_card
  WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
```

OUTPUT:

Result Grid					
Filter Rows:					
Edit:					
Export/Import:					
Wrap Cell Content:					
id	iban	pin	cvv	expiring_date	
NULL	NULL	NULL	NULL	NULL	

### SPRINT 3 - NIVEL 2 - EJERCICIO 2

Crear vista (VIEW) para Marketing con datos de compra media por compañía y otros datos como el nombre de la compañía, país, y el teléfono.

Por ello el código realiza un CREATE VIEW para definir una vista llamada 'view\_VistaMarketing' que combina datos de las tablas 'company' y 'transaction'.

La vista selecciona el nombre de la empresa, el teléfono y el país de la tabla 'company', así como el promedio redondeado de la cantidad (amount) de transacciones para cada empresa de la tabla 'transaction'.

La cláusula JOIN combina las filas de las tablas 'transaction' y 'company' en función de la igualdad de los ID de la empresa.

La cláusula GROUP BY agrupa los datos por ID de empresa para calcular el promedio de las transacciones de cada empresa.

La cláusula ORDER BY ordena los resultados en orden descendente según el promedio de compra (compra\_media).

Finalmente, se realiza una consulta para seleccionar y mostrar todos los datos de la vista 'view\_VistaMarketing', que ahora contiene la información deseada para el marketing.

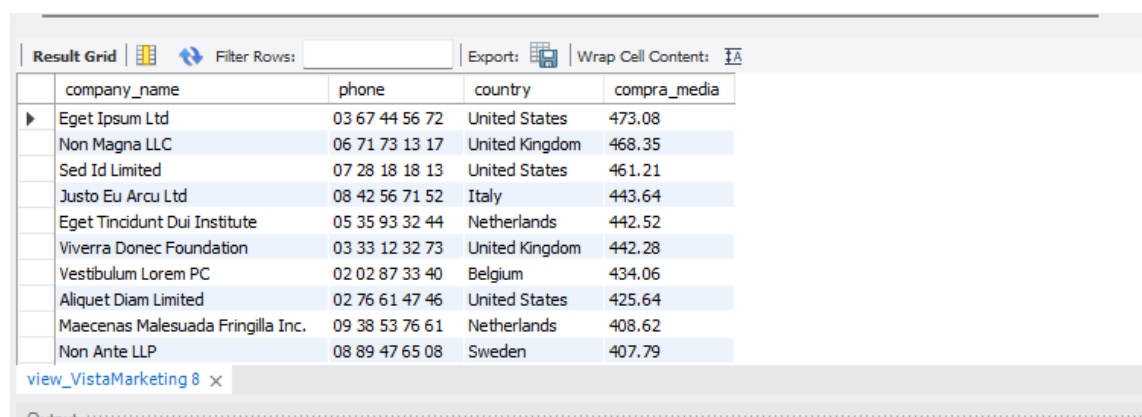
#### CÓDIGO:

```
-- SPRINT 3 - NIVEL 2 - EJERCICIO 2
-- Creación de una vista VIEW para Marketing con datos de las tablas company y transaction

-- Creación de la vista 'view_VistaMarketing' que contiene datos combinados de las tablas 'company' y 'transaction'
CREATE VIEW view_VistaMarketing AS
    SELECT c.company_name, c.phone, c.country, ROUND(AVG(amount), 2) as compra_media
    FROM transaction as t
    JOIN company as c ON c.id = t.company_id
    GROUP BY company_id
    ORDER BY compra_media desc;

-- Consulta para seleccionar y mostrar todos los datos de la vista 'view_VistaMarketing'
SELECT * FROM view_VistaMarketing;
```

#### OUTPUT:



company_name	phone	country	compra_media
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
Aliquet Diam Limited	02 76 61 47 46	United States	425.64
Maecenas Malesuada Fringilla Inc.	09 38 53 76 61	Netherlands	408.62
Non Ante LLP	08 89 47 65 08	Sweden	407.79

### **SPRINT 3 - NIVEL 2 - EJERCICIO 3**

Crear un filtro sobre la vista (VIEW) para Marketing con datos de compra media por compañía y otros datos como el nombre de la compañía, país, y el teléfono, pero tan sólo para el país: 'Germany'.

En este caso el código es simple, se utiliza una consulta SELECT para seleccionar todos los datos de la vista 'view\_VistaMarketing'.

Utiliza la cláusula FROM para especificar que los datos se tomen de la vista 'view\_VistaMarketing', utilizando el alias 'v' para referencia.

Utiliza la cláusula WHERE para filtrar las filas donde el valor de la columna 'country' (país) es 'Germany' (Alemania).

Esto devuelve todas las filas de la vista 'view\_VistaMarketing' donde el país es Alemania, lo que proporciona un filtro específico para las empresas alemanas.

#### **CÓDIGO:**

```
-- SPRINT 3 - NIVEL 2 - EJERCICIO 3
-- Usar la vista VIEW para realizar un filtro con los datos de empresas alemanas

-- Consulta para seleccionar todos los datos de la vista 'view_VistaMarketing' donde el país es 'Germany'
SELECT *
FROM view_VistaMarketing AS v
WHERE v.country = 'Germany';

#####
```

#### **OUTPUT:**

Result Grid				
Filter Rows:		Export:	Wrap Cell Content:	
company_name	phone	country	compra_media	
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	

```
659
660
661 # f)Eliminamos la columna Website de la tabla company
662 • alter table company drop column website;
663
664 # g)Modificamos el nombre del campo correo electrónico en la tabla user
665 • ALTER TABLE user
666 RENAME COLUMN email TO personal_email;
667 • describe credit_card;
668
669 # h)Modificamos tipo de datos, longitudes y datos nulos en la tabla credit_card
670 • ALTER TABLE credit_card
671 CHANGE COLUMN `iban` `iban` VARCHAR(50) NULL DEFAULT NULL ,
672 CHANGE COLUMN `pin` `pin` VARCHAR(4) NULL DEFAULT NULL ,
673 CHANGE COLUMN `cvv` `cvv` INT NULL DEFAULT NULL ,
674 CHANGE COLUMN `expiring_date` `expiring_date` VARCHAR(10) NULL DEFAULT NULL ;
675 • describe credit_card;
676
677 # i)Añadimos la columna fecha_actual a la tabla credit_card
678 • ALTER TABLE credit_card
679 ADD COLUMN `fecha_actual` DATE DEFAULT(current_date);
680 • describe transaction;
681
```



Estos son los OUTPUTS:

```
661 # f)Eliminamos la columna Website de la tabla company
662 • alter table company drop column website;
663
664 # g)Modificamos el nombre del campo correo electrónico en la tabla user
665 • ALTER TABLE user
666   RENAME COLUMN email TO personal_email;
667 • describe credit_card;
668
669 # h)Modificamos tipo de datos, longitudes y datos nulos en la tabla credit_card
670 • ALTER TABLE credit_card
671   CHANGE COLUMN `iban` `iban` VARCHAR(50) NULL DEFAULT NULL ,
672   CHANGE COLUMN `pin` `pin` VARCHAR(4) NULL DEFAULT NULL ,
673   CHANGE COLUMN `cvv` `cvv` INT NULL DEFAULT NULL ,
674   CHANGE COLUMN `expiring_date` `expiring_date` VARCHAR(10) NULL DEFAULT NULL ;
675 • describe credit_card;
676
677 # i)Añadimos la columna fecha_actual a la tabla credit_card
678 • ALTER TABLE credit_card
679   ADD COLUMN `fecha_actual` DATE DEFAULT(current_date);
680 • describe transaction;
```

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	YES		NULL	
pin	varchar(4)	YES		NULL	
cvv	int	YES		NULL	
expiring_date	varchar(10)	YES		NULL	
fecha_actual	date	YES		curdate()	DEFAULT_GENERATED

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(15)	YES	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int	YES	MUL	NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	YES		NULL	
declined	tinyint(1)	YES		NULL	

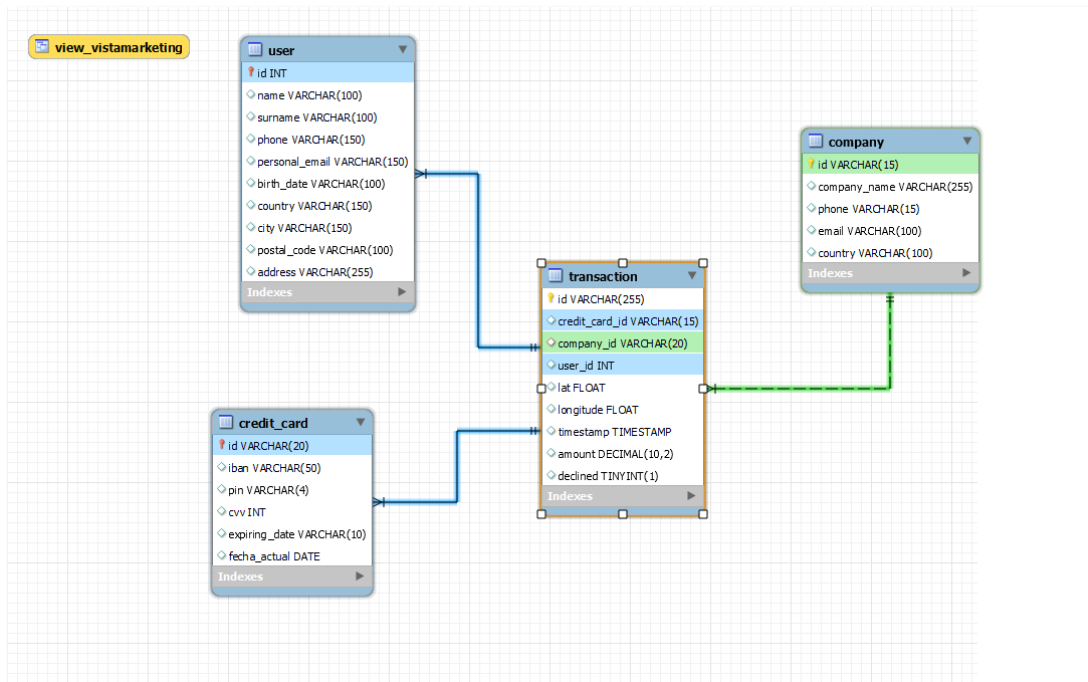
Result 16 ×

Output

Finalmente, añadimos las relaciones entidad-objeto para las claves foráneas:

```
682 # j)Añadimos las relaciones 1 a N en la tabla credit card
683 • ALTER TABLE credit_card
684 ADD CONSTRAINT `fk_credit_card`
685 FOREIGN KEY (`id`)
686 REFERENCES transaction (`credit_card_id`)
687 ON DELETE NO ACTION
688 ON UPDATE NO ACTION;
689
690
691
692
```

Este es el Output:



ATENCIÓN: PENSAMOS QUE LAS RELACIONES 1 a N están invertidas

Deberían ir en el Sentido de Transaction a company, user, y credit\_card.

### SPRINT 3 - NIVEL 3 - EJERCICIO 2

Creamos una vista (CREATE VIEW) con datos de tres tablas, es por ese motivo que debemos hacer un JOIN con las FOREIGN KEY de cada una de ella: company, transaction y credit\_card.

Desarrollamos esta idea mediante la creación de un código que crea una vista llamada view\_informe\_tecnico que combina datos de las tablas user, credit\_card, transaction, y company. Luego, realiza una consulta para seleccionar y mostrar todos los datos de esta vista.

Creamos la vista CREATE VIEW para crear una vista llamada view\_informe\_tecnico.

La vista selecciona campos específicos de las tablas transaction, company, credit\_card, y user.

Se utiliza la cláusula JOIN para combinar las filas de las tablas transaction, company, credit\_card, y user basadas en las relaciones establecidas entre ellas. Estos son los campos seleccionados

t.id: El ID de la transacción.

u.name: El nombre del usuario.

u.surname: El apellido del usuario.

d.iban: El IBAN (International Bank Account Number) de la tarjeta de crédito.

c.company\_name: El nombre de la empresa.

La cláusula ORDER BY ordena los resultados de la vista por el ID de la transacción en orden descendente (t.id desc).

Después de definir la vista, se realiza una consulta para seleccionar y mostrar todos los datos de la vista view\_informe\_tecnico. Esto mostrará el informe técnico que contiene los datos combinados de las tablas mencionadas anteriormente.

```
690 # SPRINT 3 - NIVEL 3 - EJERCICIO 2
691 # Construir una vista llamada Informe Técnico que contenga datos de las tables user y credit_card con ciertos campos
692
693 • CREATE VIEW view_informe_tecnico AS
694     SELECT t.id, u.name, u.surname, d.iban, c.company_name
695     FROM transaction as t
696     JOIN company as c on c.id = t.company_id
697     JOIN credit_card as d on d.id = t.credit_card_id
698     JOIN user as u on u.id = t.user_id
699     ORDER BY t.id desc;
700
701 • select * from view_informe_tecnico
702
703
704
```

id	name	surname	iban	company_name
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E46468	Molly	Gilliam	SE2813123487163628531121	Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incorporated
FD89D518-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	Neque Tellus Imperdiet Corp.
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	Hakeem	Alford	MD1234119525145401270486	Nunc Interdum Incorporated
FBD7E0D6-BA68-F5BC-0CA9-EA4B8760100C	Hedwig	Gilbert	MU4132333444534342541344788855	Mauris Id Inc.
FAC76&Rn-R44&R-F9&A&R-FR97-476C7F17671C	Slade	Poola	MT05TWCF5886R20N575771634583813	Arquill P

view\_informe\_tecnico 19 x