Sprint 3

Describimos las tablas que se van a crear en este proyecto de manipulación de datos.

Tabla Credit card

Id: varchar (20) caracteres, define un identificador único (ID) el cual no debe repetirse en todo el documento.

Iban: varchar (50) caracteres, IBAN de una cuenta (código internacional de cuenta bancaria) es un código acordado de forma internacional de hasta 34 dígitos y caracteres alfanuméricos que ayuda a los bancos a procesar transferencias internacionales.

Pin: varchar (4) caracteres, **PIN** es el acrónimo de «Personal Identification Number» (número de identificación personal en español) y se refiere a un número de seguridad que se utiliza para autenticar a una persona en un sistema. Es un **código numérico** que generalmente consta de cuatro o seis dígitos.

cvv: int (número entero) El código CVV o CVC es un grupo de 3 o 4 números situado en el reverso de la tarjeta de crédito o débito. Dicho código se utiliza como método de seguridad en transacciones en las que la tarjeta no está físicamente presente, como en compras por teléfono o internet.

Tabla Data_user

id: INT identificador único de la persona usuario de la tabla. (en este caso el id, se relaciona con user_id de la columna transaction que también es int por lo cual hay consistencia en los datos)

name: varchar (100) campo de texto, nombre del usuario.

surname: Varchar (100) campo de texto, apellido del usuario.

Phone: Varchar (150), campo de texto que identifica el número de teléfono en diferentes formatos, incluyendo el internacional.

personal_email: Varchar (150), cadena de texto y caracteres, dirección de correo electrónico.

birth_date: Varchar (100) atributo que almacena información del día, mes y año, para esta tabla esta creada de tipo varchar y lo dejaremos de esta manera para respetar la consistencia de los datos. Pero si se quiere hacer transformaciones en el formato es mejor de tipo Date.

country: Varchar (150), cadena de texto, define el nombre de un país.

city: Varchar (150), cadena de texto, define el nombre de una ciudad.

postal_code: Varchar (100) cadena de texto, representa el código postal del usuario.

address: Varchar (255) cadena de texto, representa la dirección del usuario.

Nivel 1

Ejercicio 1

Tu tarea es diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de forma única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "company"). Después de crear la tabla será necesario que ingreses la información del documento denominado "datos_introducir_credit". Recuerda mostrar el diagrama y realizar una breve descripción del mismo.

```
3  • CREATE TABLE credit_card (
4      id VARCHAR(20) PRIMARY KEY,
5      iban VARCHAR(50),
6      pan CHAR(30),
7      pin CHAR(4),
8      cvv INT,
9      expiring_date VARCHAR(20)
10 );
11
12 • select *
13   from transactions.credit_card;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	iban	pan	pin	cvv	expiring_date
CcS-4857	XX485759183529250585077 1	2314242385113924	1819	467	09/27/25
CcS-4858	XX858176813700243609402 5	6582720299715533	3964	817	12/28/28
CcS-4859	XX782693049142355360937 0	8861684536289642	4983	277	11/26/26
CcS-4860	XX555959036883530464529 9	2481155515498459	6876	661	07/27/27
CcS-4861	XX203518287719519162730 7	1308930301149557	5710	398	04/25/26
CcS-4862	XX477472146246364540975 8	6715617009807829	4042	174	11/27/26

credit_card 230

Output

Action Output

#	Time	Action	Message
4999	11:38:24	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9578', 'XX991539646...	1 row(s) affected
5000	11:38:24	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9579', 'XX296393091...	1 row(s) affected
5001	11:38:24	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9580', 'XX781258889...	1 row(s) affected
5002	11:38:24	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9581', 'XX915670516...	1 row(s) affected
5003	11:41:17	select * from transactions.credit_card	5000 row(s) returned

Se ha creado la tabla credit_card

```
15 • alter table transaction
16       ADD foreign key (credit_card_id) references credit_card(id);
17
18
```

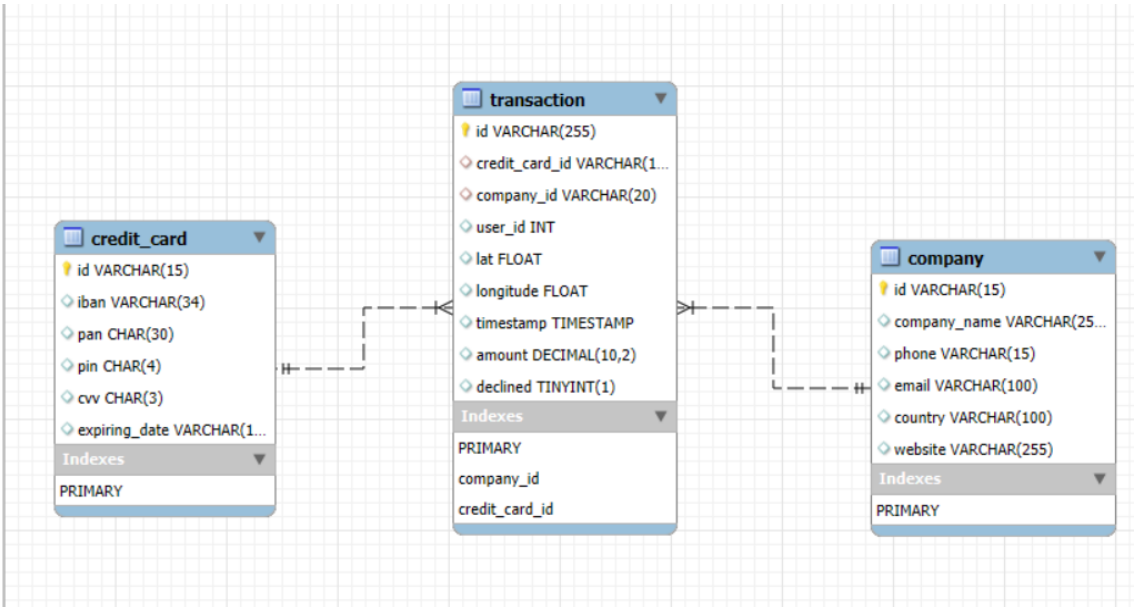
Output

Action Output

#	Time	Action	Message
1	23:15:31	alter table transaction ADD foreign key (credit_card_id) references credit_card(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

Se modificará la tabla transacción, credit_card_id será una clave foránea y esta clave debe coincidir con la clave id de la tabla credit_card.

Ahora se ve así el diagrama.

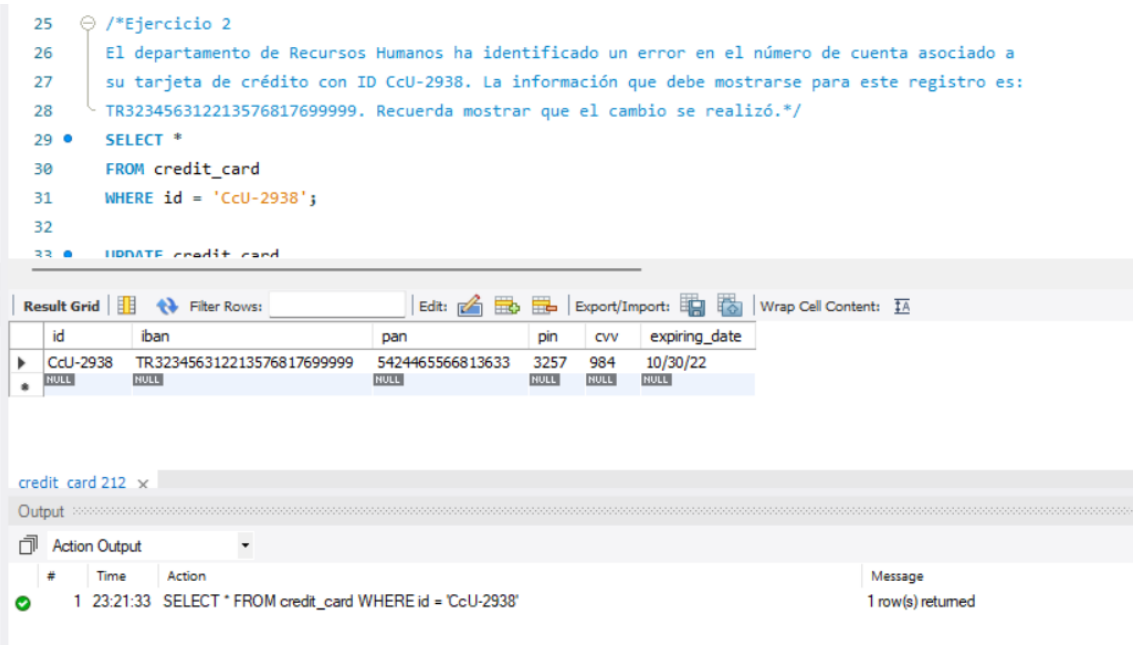


Cada transaction:

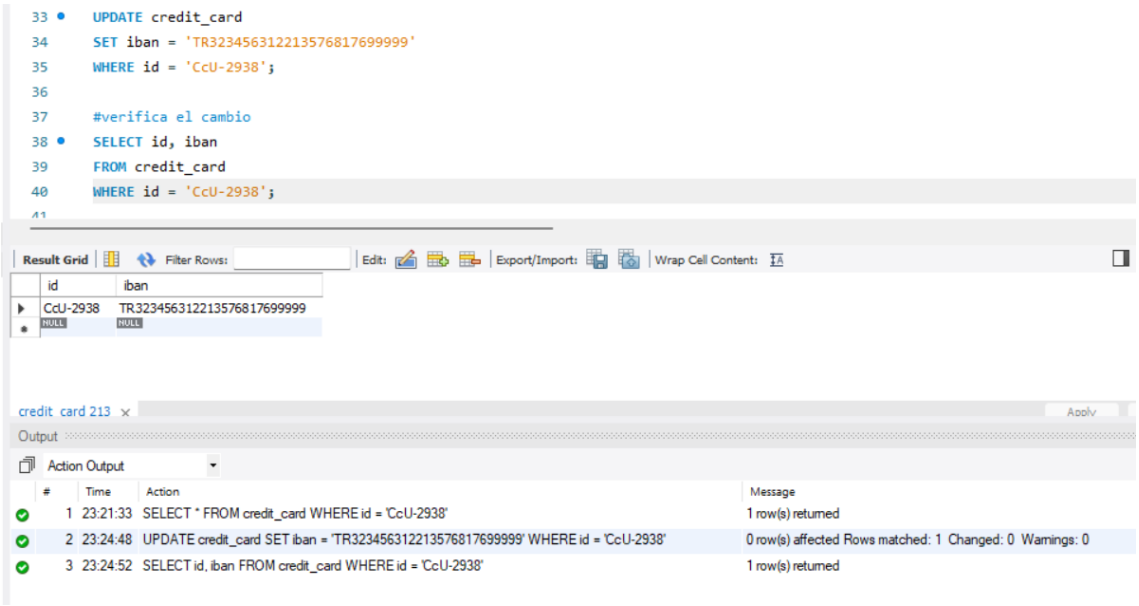
está asociada a una compañía y a una tarjeta de crédito.

Cada credit_card y company:

pueden tener múltiples transacciones asociadas.



En el ejercicio 2 miramos entonces que hay en la columna que asociada al número.



Realizamos el cambio con un update y luego verificamos que se haya realizado.

Hay un valor en la columna company que no existe.

Probaremos insertando datos a company(id)

Insertamos y comprobamos

```
INSERT INTO company(id
      )
VALUES('b-9999'
    );


SELECT *
FROM company
WHERE id=('b-9999');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

company 51 ×

Output

Action Output

#	Time	Action	Message
1	12:35:33	INSERT INTO company(id) VALUES('b-9999')	1 row(s) affected
2	12:35:38	SELECT * FROM company WHERE id=('b-9999')	1 row(s) returned

Ahora debemos insertar el company_id para poder insertar los demás datos

```
INSERT INTO credit_card(id)
VALUES('CcU-9999');

SELECT *
FROM credit_card
WHERE id=('CcU-9999');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

credit_card 55 ×

Output

Action Output

#	Time	Action	Message
1	12:43:13	SELECT * FROM company WHERE id=('b-9999')	1 row(s) returned
2	12:47:28	INSERT INTO credit_card(id) VALUES('CcU-9999')	1 row(s) affected
3	12:47:32	SELECT * FROM credit_card WHERE id=('b-9999')	0 row(s) returned
4	12:48:03	SELECT * FROM credit_card WHERE id=('CcU-9999')	1 row(s) returned

Insertamos el credit_card_id, para continuar agregando los datos

```sql
INSERT INTO transaction (id,credit_card_id,company_id,user_id,lat,longitude,amount,declined)
VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD','CcU-9999','b-9999',9999,829.999,-117.999,111.11,0);

SELECT *
FROM transaction
WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';



/*SET foreign_key_checks =0;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

transaction 56 | Apply

Output

Action Output

#	Time	Action	Message
1	12:51:36	INSERT INTO transaction (id,credit_card_id,company_id,user_id,lat,longitude,amount,declined) VALUE...	1 row(s) affected
2	12:51:40	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD'	1 row(s) returned

Finalmente agregamos los datos a la tabla y comprobamos que se hayan agregado correctamente.

```sql
# **Exercici 4**
/*de la tabla credit_card. Recuerda mostrar el cambio realizado.
ALTER TABLE credit_card DROP COLUMN pan;*/

ALTER TABLE credit_card DROP COLUMN pan;

SELECT *
FROM credit_card;



/*Nivel 2
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	iban	pin	cvv	expiring_date
CcS-4857	XX485759183529250585077 1	1819	467	09/27/25
CcS-4858	XX858176813700243609402 5	3964	817	12/28/28
CcS-4859	XX782693049142355360937 0	4983	277	11/26/26
CcS-4860	XX555959036883530464529 9	6876	661	07/27/27
CcS-4861	XX203518287719519162730 7	5710	398	04/25/26

credit_card 312 | Apply

Output

Action Output

#	Time	Action	Message
105128	22:54:36	SET foreign_key_checks =1	0 row(s) affected
105129	22:54:39	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD'	1 row(s) returned
105130	22:56:25	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
105131	22:56:31	SELECT * FROM credit_card	5000 row(s) returned

eliminamos la columna pan y luego comprobamos que ya no aparece en la tabla.

```
/*Nivel 2
Ejercicio 1
Elimina de la tabla transacción el registro con ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de datos.*/
DELETE FROM transactions.transaction
WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';

SELECT *
FROM transaction
WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

transaction 221

Output

Action Output

#	Time	Action	Message
1	00:16:08	DELETE FROM transactions.transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	0 row(s) affected
2	00:16:11	SELECT * FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	0 row(s) returned

Eliminamos el registro del WHERE id =000447FE-B650-4DCF-85DE-C7ED0EE1CAAD, luego comprobamos que el cambio se hay realizado, hemos comprobado porque luego toda la columna dice NULL.

```
/*Ejercicio 2
La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias.
Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones.
 menor promedio de compra.*/
```

```
/*Ejercicio 2
La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias.
Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones.
 menor promedio de compra.*/
CREATE VIEW transactions.VistaMarketing AS
SELECT c.phone as CONTACT_NUMBER,
 c.company_name AS COMPANY,
 c.country AS COUNTRY,
 ROUND(AVG(t.amount),2) AS AVERAGE_SALES
FROM company c
JOIN transaction t ON c.id = t.company_id
GROUP BY c.id,c.company_name, c.country, c.phone
ORDER BY AVERAGE_SALES;
```

Output

Action Output

#	Time	Action	Message
1	00:30:26	CREATE VIEW transactions.VistaMarketing AS SELECT c.phone as CONTACT_NUMBER, c.company_...	0 row(s) affected

Ahora crear la vista

Comprobamos que la vista se ha creado.

```
117 •  CREATE VIEW transactions.VistaMarketing AS
118    SELECT  c.company_name AS COMPANY,
119    c.phone as CONTACT_NUMBER,
120    c.country AS COUNTRY,
121    ROUND(AVG(t.amount),2) AS AVERAGE_SALES
122    FROM company c
123    JOIN transaction t ON c.id = t.company_id
124    GROUP BY c.id, c.company_name, c.country, c.phone
125    ORDER BY AVERAGE_SALES DESC;
126
127 •  SELECT *
128    FROM transactions.vistamarketing;
129
```

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

COMPANY	CONTACT_NUMBER	COUNTRY	AVERAGE_SALES
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Pretium Neque Corp.	07 77 48 55 28	Australia	276.16
Urna Convallis Associates	06 01 24 77 04	United States	274.24
At Associates	09 56 61 10 65	New Zealand	272.21
Metus Vitae Associates	08 25 44 40 66	Australia	270.08
Aliquet Diam Limited	02 76 61 47 46	United States	269.60

vistamarketing 142 ×

Output

Action Output

#	Time	Action	Message
5047	15:39:44	CREATE VIEW transactions.VistaMarketing AS SELECT c.company_name AS COMPANY, c.phone ...	0 row(s) affected
5048	15:39:47	SELECT * FROM transactions.vistamarketing	101 row(s) returned

Miramos como se ve la vistamarketing.

```sql
/*ejercicio 3
Filtra la vista VistaMarketing para mostrar sólo las compañías que tienen su país de residencia en "Germany"*/
SELECT *
FROM transactions.VistaMarketing
WHERE COUNTRY= 'Germany';
```

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

CONTACT_NUMBER	COMPANY	COUNTRY	AVERAGE_SALES
01 45 73 52 16	Aliquam PC	Germany	253.14
06 88 43 15 63	Augue Foundation	Germany	253.51
05 62 87 14 41	Auctor Mauris Corp.	Germany	254.77
02 66 31 61 09	Rutrum Non Inc.	Germany	255.14
09 34 65 40 60	Ac Industries	Germany	255.15

VistaMarketing 226

Output

Action Output

#	Time	Action	Message
1	00:35:42	SELECT * FROM transactions.VistaMarketing WHERE COUNTRY= 'Germany'	8 row(s) returned

Filtramos la vista por germany.

Nivel 3.

```sql
/*Nivel 3
Ejercicio 1*/
/*La próxima semana tendrás una nueva reunión con los gerentes de marketing.
Un compañero de tu equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. */
```

CREAR la tabla users

```sql
CREATE TABLE IF NOT EXISTS user (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(100),
    city VARCHAR(100),
    postal_code VARCHAR(20),
    address VARCHAR(255),
    FOREIGN KEY (id) REFERENCES transaction(user_id)
);
#Comprobando la tabla
SELECT *
FROM transactions.user;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York	10001	348-7818 Sagittis St.
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Philadelphia	19101	903 Sit Ave
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Houston	77001	736-2063 Tellus St.
4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Phoenix	85001	Ap #545-2244 Erat. Rd.
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Philadelphia	19101	341-2821 Ultrices Av.

user 316 | Apply

Output

Action Output

#	Time	Action	Message
110138	23:01:35	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) ...	1 row(s) affected
110139	23:01:54	SELECT * FROM transactions.user	5000 row(s) returned

Comprobamos que se haya creado correctamente la tabla users.

```
174     #user: cambiar nombre de la tabla user a data_user
175 •   ALTER TABLE user RENAME  to data_user;
176
177     #user: cambiar nombre  a data_user, canbia email a personal_email
178 •   ALTER TABLE data_user RENAME COLUMN email to personal_email;
179
180     #company: elimina columna website
181 •   ALTER TABLE company DROP COLUMN website;
182
183     #credit_card: AGREGAR columna fecha_actual DATE
184 •   ALTER TABLE  credit_card ADD COLUMN fecha_actual DATE;
185
186
187
188
189
190
191
192
```

Output

Action Output

#	Time	Action	Message
1	10:52:28	ALTER TABLE user RENAME to data_user	0 row(s) affected
2	10:52:32	ALTER TABLE data_user RENAME COLUMN email to personal_email	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	10:52:35	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
4	10:52:38	ALTER TABLE credit_card ADD COLUMN fecha_actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Luego realizamos cambios como renombrar la columna user a data_user, cambiar el nombre del email por personal_email, eliminar la columna website de la tabla company por último agregar la columna Date a credit_card.

```
/*Ejercicio 2
La empresa también le pide crear una vista llamada "InformeTecnico" que contenga la siguiente información:

ID de la transacción
Nombre del usuario/a
Apellido del usuario/a
IBAN de la tarjeta de crédito usada.
Nombre de la compañía de la transacción realizada.
Asegúrese de incluir información relevante de las tablas que conocerá y utilice alias para cambiar
de nombre columnas según sea necesario.
Muestra los resultados de la vista, ordena los resultados de forma descendente en función de la variable ID de transacción.*/
```

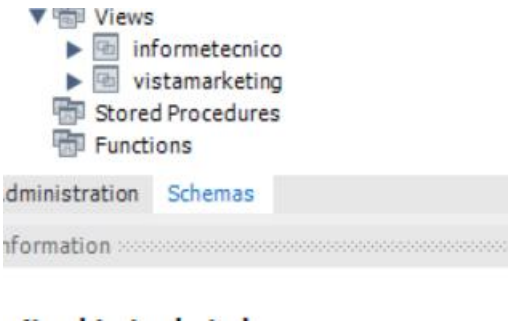
Creamos la vista

```sql
CREATE VIEW transactions.InformeTecnico AS
SELECT
t.id AS ID_Transaccions,
u.name AS USER_NAMES,
u.surname AS USERS_LASTNAME,
cc.iban AS IBAN,
c.company_name AS COMPANY_NAME
FROM transaction AS t
JOIN data_user AS u ON t.user_id=u.id
JOIN credit_card AS cc ON t.credit_card_id = cc.id
JOIN company AS c ON t.company_id = c.id
ORDER BY t.id DESC;
```

Output

Action Output

#	Time	Action	Message
1	11:06:00	CREATE VIEW transactions.InformeTecnico AS SELECT t.id AS ID_Transaccions, u.name AS USER_...	0 row(s) affected



verificamos que se haya creado la vista.

Ahora podemos ver

```sql
SELECT * FROM InformeTecnico;
```

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

ID_Transaccions	USER_NAMES	USERS_LASTNAME	IBAN	COMPANY_NAME
FFFD31D6-9495-47CE-B54A-7DB8E1CC274B	Bmrgli	Tprvvmrc	XX7948144512112891824909 22	Turpis Company
FFFCF76D-ECF0-4985-A2D0-B2A7B75998FC	Dfrled	Vilqcjdl	XX6362517016478920366760 34	Amet Nulla Donec Corporation
FFFC9E8D-27C7-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	XX1626771433042236314375 67	Nunc Interdum Incorporated
FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzjpa	Uirzjulh	XX3951142670820199525670 52	Viverra Donec Foundation
FFF9E3CE-234E-408C-A8EF-F9CAD577224A	Yshimq	Zpsjsleed	XX8845462156537570367941	Convallis In Incorporated
FFF9E178-6CD2-4DF9-99B0-49AE068809B1	Jevepx	Xwcwzwnm	XX3214055157116543847114 81	Mus Aenean Eget Foundation

Result Grid | Form Editor

InformeTecnico 229 | Read Only

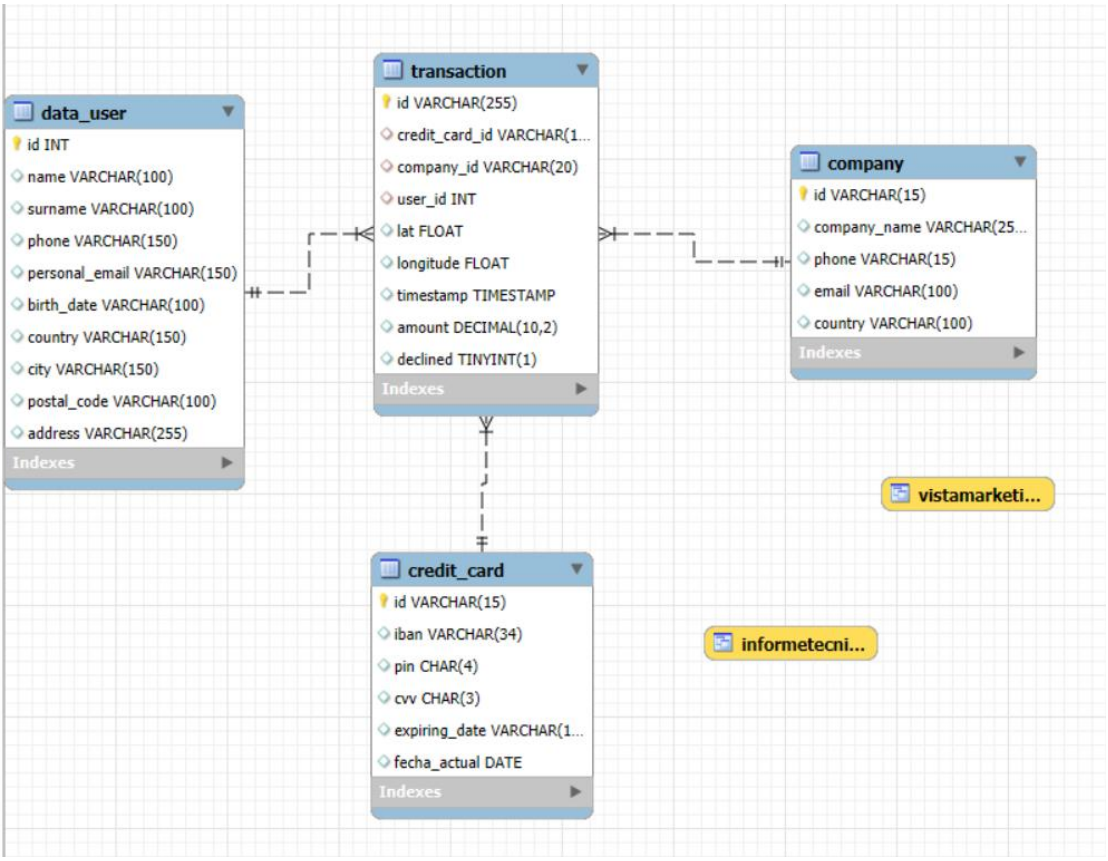
Output

Action Output

#	Time	Action	Message
1	11:06:00	CREATE VIEW transactions.InformeTecnico AS SELECT t.id AS ID_Transaccions, u.name AS USER_...	0 row(s) affected
2	11:08:05	SELECT * FROM InformeTecnico	99999 row(s) returned

la vista de informe técnico.

Finalmente, el diagrama ha quedado de la siguiente manera:



Transacción a data_user

Relación de uno a muchos

Un usuario puede tener muchas transacciones y cada transacción esta hecha por un usuario.

Transacción a company

Relación de uno a muchos.

Una empresa puede tener muchas transacciones y cada transacción pertenece a una empresa

Transacción a credit_card

Relación de uno a muchos.

La tarjeta de crédito puede realizar muchas transacciones y cada transacción pertenece a una tarjeta de crédito.

Así que la tabla principal es la tabla transacciones.