SPRINT S3.01 – MANIPULACIÓN DE TABLAS

NIVEL 1

Ejercicio 1.1

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 ingresar la información del documento denominado "datos_introducir_credit". Recordar mostrar el diagrama y realizar una breve descripción del mismo.

Creando la tabla **"credit_card"** siguiendo las indicaciones y estableciendo el campo **"id"** como primary key además de definir la relación con la tabla **transaction** a través de la **foreign key** desde la creación.

```
use transactions;
create table credit_card (
    id varchar(15) unique,
    iban varchar(50),
    pan varchar(50),
    pin int,
    cvv int,
    expiring_date varchar(10),
    primary key (id),
    foreign key(id) references transaction(credit_card_id)
    );
```

Nos arroja el siguiente error:

Por ello primero procedemos a crear el siguiente **indice**:

```
create index idx_creditcard
on transaction(credit_card_id);
```

Luego de ello, corremos nuevamente el código inicial para crear la tabla, la cual conseguiremos realizarla con éxito, como se observa:

```
# Ejercicio 1.1 : Creando la tabla credit_card siguiendo las indicaciones y estableciendo el campo "id" como primary key.

use transactions;
create table credit_card (
    id varchar(15) unique,
    iban varchar(50),
    pan varchar(50),
    pin int,
    cvv int,
    expiring_date varchar(10),
    primary key (id),
    foreign key(id) references transaction(credit_card_id)
    );

describe credit_card;
```

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

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
iban	varchar(50)	YES		NULL	
pan	varchar(50)	YES		NULL	
pin	int	YES		NULL	
cvv	int	YES		NULL	
expiring_date	varchar(10)	YES		NULL	

Result 21

Output

Action Output

#	Time	Action	Message
106	00:08:34	create table credit_card (id varchar(15) unique, iban varchar(50), pan varc...	0 row(s) affected
107	00:09:07	describe credit_card	6 row(s) returned

```
show create table credit_card;
```

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

Table	Create Table
credit_card	CREATE TABLE `credit_card` (`id` varchar(15) NOT NULL, `iba

```
CREATE TABLE `credit_card` (
  `id` varchar(15) NOT NULL,
  `iban` varchar(50) DEFAULT NULL,
  `pan` varchar(50) DEFAULT NULL,
  `pin` int DEFAULT NULL,
  `cvv` int DEFAULT NULL,
  `expiring_date` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`),
  CONSTRAINT `credit_card_ibfk_1` FOREIGN KEY (`id`) REFERENCES `transaction` (`credit_card_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

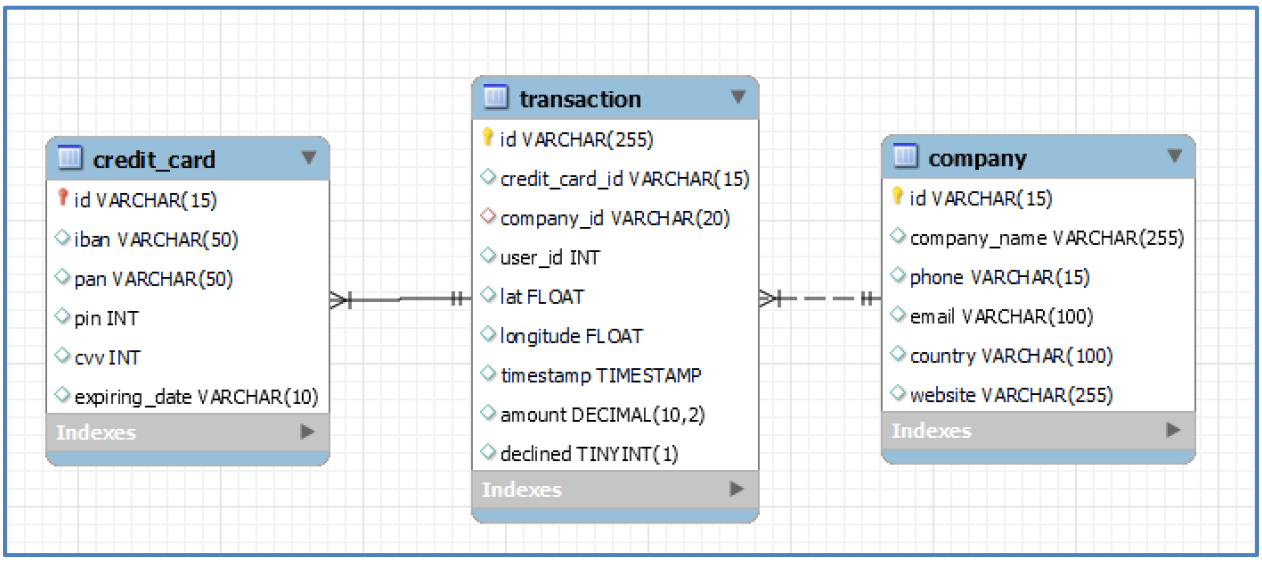
Result 22

Output

Action Output

#	Time	Action	Message
107	00:09:07	describe credit_card	6 row(s) returned
108	00:11:28	show create table credit_card	1 row(s) returned

Una vez cargados los datos en nuestra nueva tabla tenemos ya las 3 tablas relacionadas, como mostramos a continuación:



La base de datos **`transactions`** ahora cuenta con 3 tablas, siendo creada a su vez la relación de 1 a muchos entre la tabla **`credit_card`** y **`transaction`** a través de la foreign key en **`transaction (credit_car_id)`**.

Ejercicio 1.2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938. La información que debe mostrarse para este registro es: R323456312213576817699999. Recordar mostrar que el cambio se realizó.

Debemos corregir:

```
312     # Ejercicio 1.2 : Hay un error que debemos corregir al usuario con ID: CcU-2938
313
314 •   SELECT *
315     FROM credit_Card
316     WHERE id = 'CcU-2938';
317
318
```

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

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

credit_Card 2

Output

Action Output

#	Time	Action	Message
1	01:13:59	SELECT id, iban FROM credit_Card WHERE id = 'CcU-2938' LIMIT 0, 50000	1 row(s) returned
2	01:14:31	SELECT * FROM credit_Card WHERE id = 'CcU-2938' LIMIT 0, 50000	1 row(s) returned

Actualizando (corrigiendo) el `iban` anterior por el `iban`: **"R323456312213576817699999"** obtendremos:

```
318
319     # Corregimos el iban:
320
321 •   UPDATE credit_card
322     SET iban = 'R323456312213576817699999'
323     WHERE id = 'CcU-2938';
324
325     # Comprobamos la actualización:
326
327 •   SELECT *
328     FROM credit_Card
329     WHERE id = 'CcU-2938';
```

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

credit_Card 3

Output

Action Output

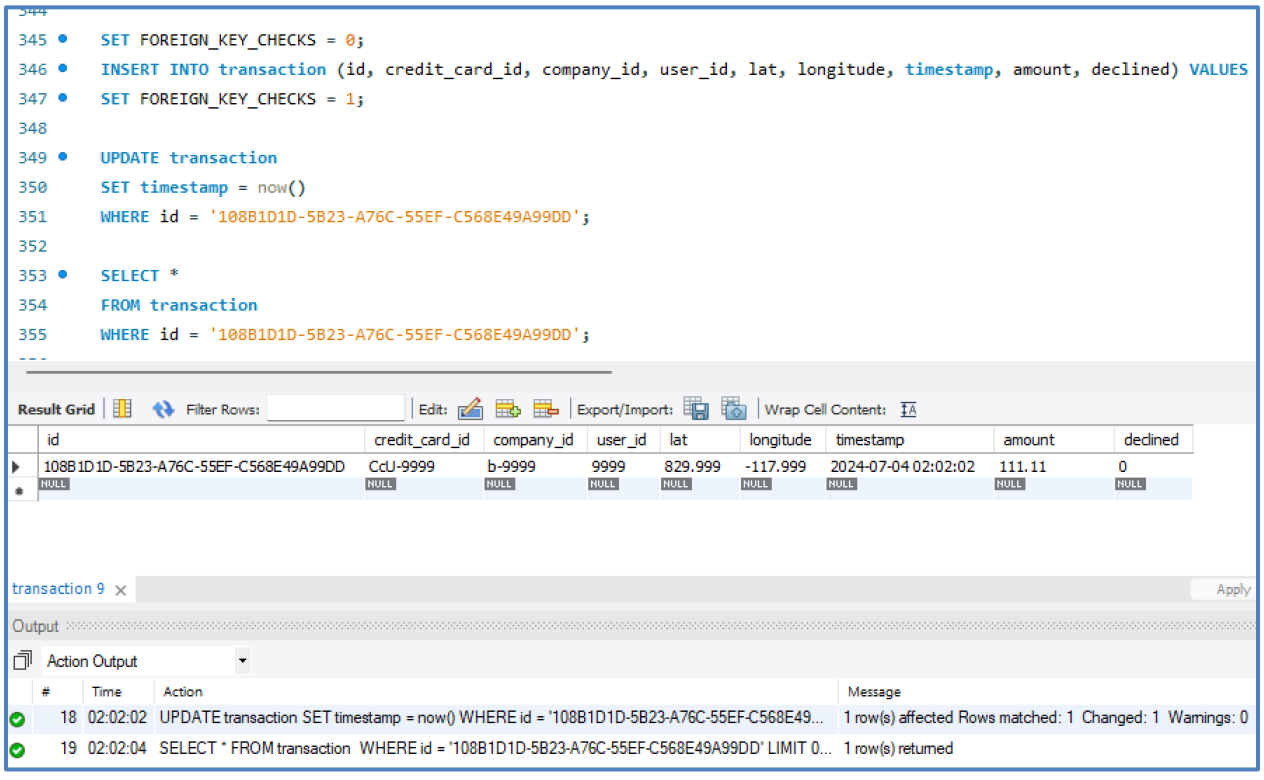
#	Time	Action	Message
3	01:18:05	update credit_card set iban = 'R323456312213576817699999' where id = 'CcU-2938'	1 row(s) affected Rows matched: 1 Changed: 1
4	01:18:51	SELECT * FROM credit_Card WHERE id = 'CcU-2938' LIMIT 0, 50000	1 row(s) returned

Ejercicio 1.3

En la tabla “transaction” ingresar un nuevo usuario con la siguiente información:

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lato	829.999
longitud	-117.999
amunt	111.11
declined	0

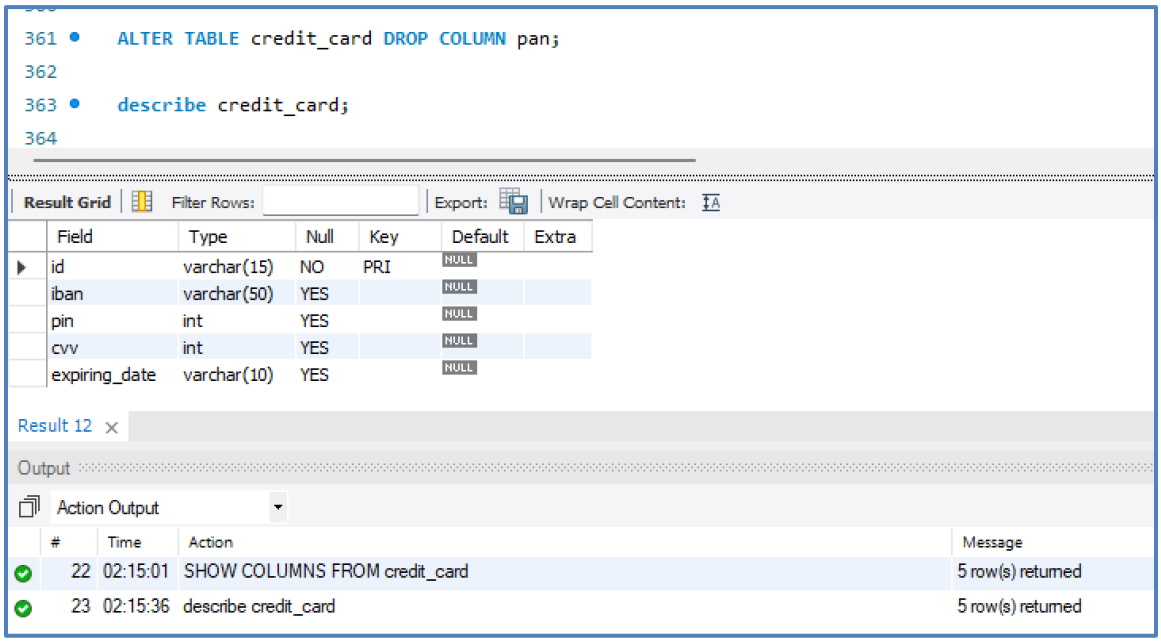
Debemos desactivar el modo seguro de actualización para luego introducir el nuevo registro proporcionado, y nuevamente activar el modo seguro de actualización. Seguidamente actualizamos la tabla y comprobamos que el nuevo registro fue introducido correctamente.



Ejercicio 1.4

Desde recursos humanos solicitan eliminar la columna "pan" de la tabla credit_card. Recordar mostrar el cambio realizado.

Borramos la columna "**pan**" y mostramos como esta ya no aparece en la descripción de la tabla **credit_card**.

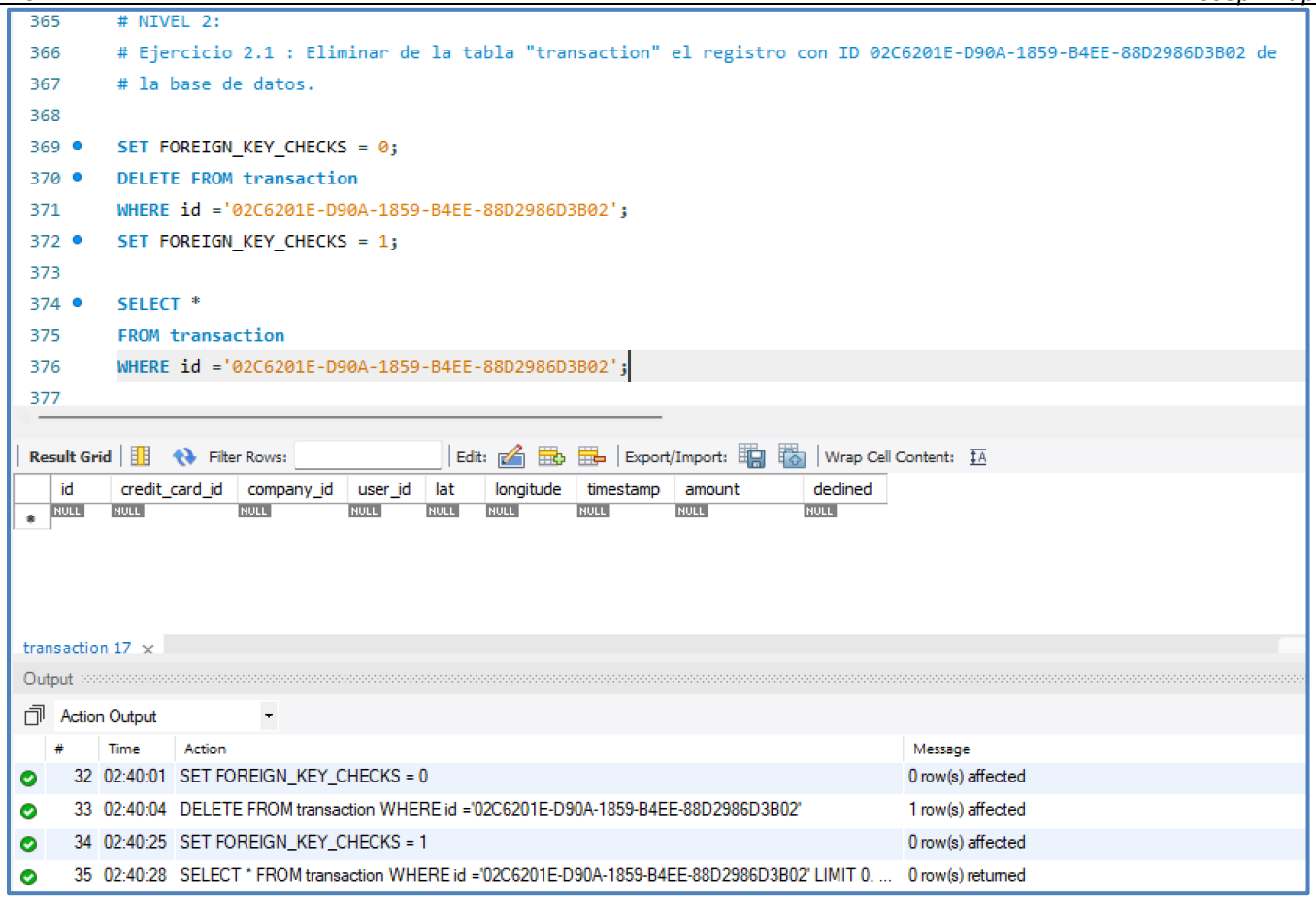


NIVEL 2

Ejercicio 2.1

Eliminar de la tabla "transaction" el registro con ID "02C6201E-D90A-1859-B4EE-88D2986D3B02" de la base de datos.

Eliminaremos el registro indicado con los comandos mostrados en la imagen siguiente:



Ejercicio 2.2

La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones. Será necesario crear una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía, Teléfono de contacto, País de residencia, media de compra realizado por cada compañía. Presentar la vista creada, ordenando los datos de mayor a menor promedio de compra.

La query solicitada entraría dentro de la categoría de vista compleja dado que utilizariamos sentencias como **`join`**, **`group by`**, etc. Al mismo tiempo, considerando solo las compras (declined = 0) efectuadas y redondeando y ordenando por los valores promedios, obtenemos el siguiente resultado:

```
381
382     # Ejercicio 2.2 : Creando una vista compleja denominada "VistaMarketing".
383
384 •   CREATE VIEW VistaMarketing AS
385     SELECT company_name, phone, country, ROUND(AVG(amount), 2) AS Media_Compra_Compañia
386     FROM company
387     JOIN transaction ON company.id = transaction.company_id
388     where declined = 0
389     GROUP BY company_name, phone, country
390     ORDER BY Media_Compra_Compañia DESC;
391
392 •   SELECT *
393     FROM VistaMarketing;
```

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

company_name	phone	country	Media_Compra_Compañia
Eget Ipsum Ltd	03 67 44 56 72	United States	481.86
Sed Id Limited	07 28 18 18 13	United States	477.51
Neque Tellus Incorporated	04 43 18 34 19	Ireland	477.10
Nunc Sit Incorporated	07 28 42 63 63	Norway	461.83
Non Magna LLC	06 71 73 13 17	United Kingdom	458.74
Maecenas Malesuada Fringilla Inc.	09 38 53 76 61	Netherlands	451.29
Erat LLP	03 18 88 77 79	Netherlands	448.44
Tortor Nunc Commodo Company	05 35 92 77 16	United States	447.11
Justo Eu Arcu Ltd	08 42 56 71 52	Italy	444.16

VistaMarketing 22

Output

Action Output

#	Time	Action	Message
50	03:21:54	CREATE VIEW VistaMarketing AS SELECT company_name, phone, country, ROUND(AVG(amoun...	0 row(s) affected
51	03:21:55	SELECT * FROM VistaMarketing LIMIT 0, 50000	100 row(s) returned

Ejercicio 2.3

Filtrar la vista "VistaMarketing" para mostrar solo las compañías que tienen su país de residencia en Alemania.

Aplicaremos el filtro mediante el uso de la clausula **`WHERE`** a la vista ya creada. En la imagen apreciamos el script y su resultado.

```
390
391     # Ejercicio 2.3 : Filtrando la "VistaMarketing" para mostrar solo las compañias con sede en "Germany"
392
393     SELECT *
394     FROM VistaMarketing
395     WHERE country ='Germany';
```

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

company_name	phone	country	Media_Compra_Compañia
Ac Industries	09 34 65 40 60	Germany	396.15
Auctor Mauris Corp.	05 62 87 14 41	Germany	308.99
Ac Fermentum Incorporated	06 85 56 52 33	Germany	293.57
Aliquam PC	01 45 73 52 16	Germany	280.34
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90
Nunc Interdum Incorporated	05 18 15 48 13	Germany	242.95
Convallis In Incorporated	06 66 57 29 50	Germany	60.99
Augue Foundation	06 88 43 15 63	Germany	15.05

VistaMarketing 23

Output

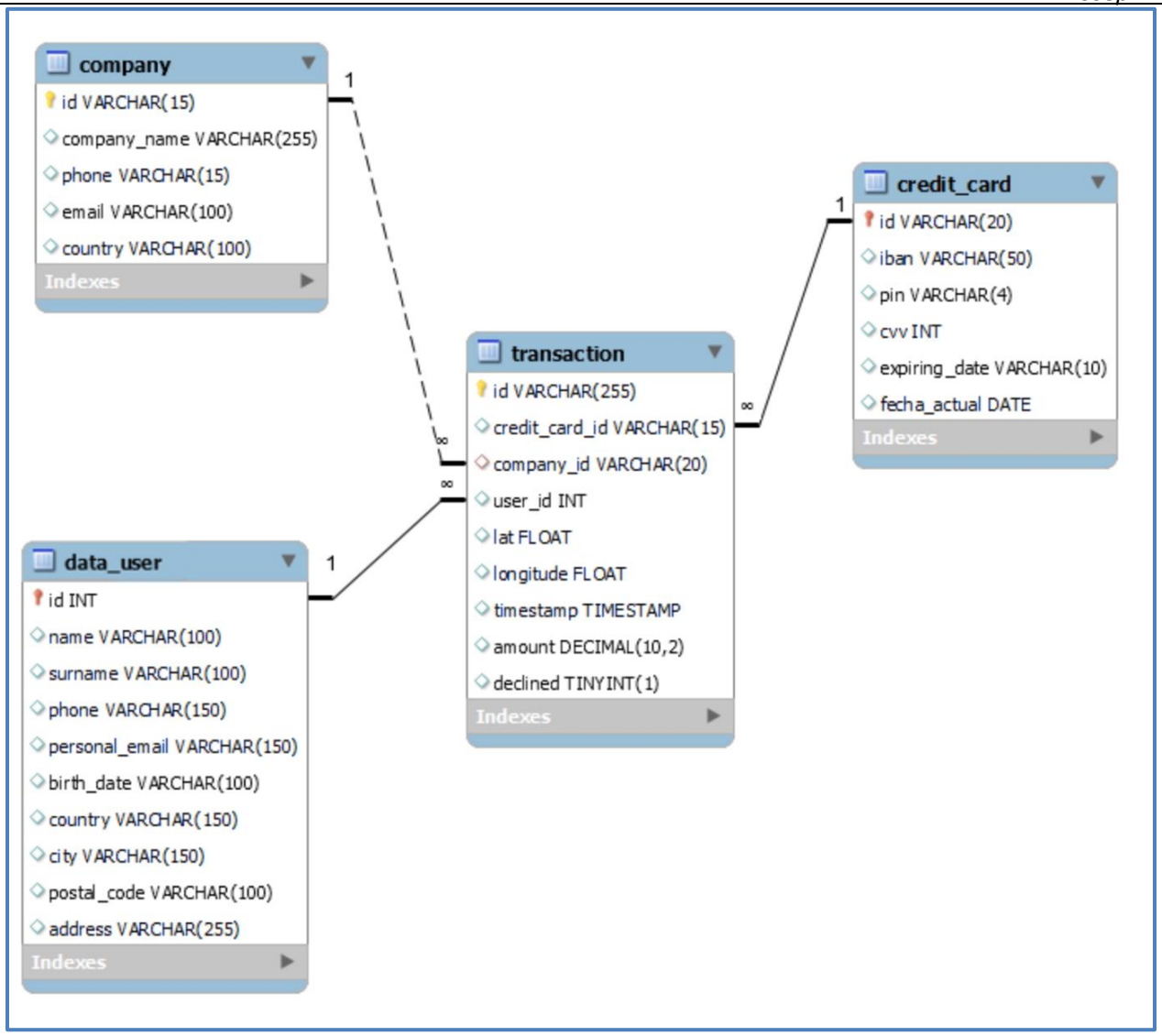
Action Output

#	Time	Action	Message
51	03:21:55	SELECT * FROM VistaMarketing LIMIT 0, 50000	100 row(s) returned
52	03:30:37	SELECT * FROM VistaMarketing WHERE country ='Germany' LIMIT 0, 50000	8 row(s) returned

NIVEL 3

Ejercicio 3.1

La próxima semana se tendrá una nueva reunión con los gerentes de marketing. Un compañero del equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Pide ayudarle a dejar los comandos ejecutados para obtener el siguiente diagrama:



Para esta actividad es necesario describir el "paso a paso" de las tareas realizadas. Es importante realizar descripciones sencillas, simples y fáciles de comprender. Para realizar esta actividad se debe trabajar con los archivos denominados "estructura_datos_user" y "datos_introducir_user".

Observando el diagrama de relación de tablas podemos notar que hay una serie de comandos que se han ejecutado para llegar a tal situación y los describiremos en los pasos siguientes.

 # creación la tabla "user" y subida de datos

Paso 1: Crear la tabla data_user con los comandos mostrados en la imagen siguiente:

```
405     #Paso 1: Se ha creado la tabla "user" siguiendo la instrucción del script "estructura_datos_user":
406
407 •   create index idx_user_id on transaction(user_id);
408
409 •   create table if not exists user (
410         id INT primary key,
411         name varchar(100),
412         surname varchar(100),
413         phone varchar(150),
414         email varchar(150),
415         birth_date varchar(100),
416         country varchar(150),
417         city varchar(150),
418         postal_code varchar(100),
419         address varchar(255),
420         foreign key(id) references transaction(user_id)
421         );
```

Paso 2: Se carga toda la data del archivo **datos_introducir_user**

 # Alteraciones en la tabla "company":

Paso 3: Se elimina el campo `website` de la tabla `company`

```
712 •   alter table company
713      drop website;
714
```

 # Alteraciones en la tabla "credit_card":

Paso 4: Se modifica la longitud del tipo de dato del campo id de la tabla `credit_card`, de `varchar(15)` a `varchar(20)`

```
717 •   alter table credit_card
718     modify column id varchar(20);
```

Paso 5: Se modifica el tipo de dato del campo `pin` de la tabla `credit_card`, de `int` a `varchar(4)`

```
722 •   alter table credit_card
723     modify pin varchar(4);
```

Paso 6: Se crea el campo `fecha_actual` siendo del tipo `DATE`

```
727 •   alter table credit_card
728     add column fecha_actual date;
```

 # Alteraciones en la tabla "user":

Paso 7: En la tabla `user`, se modifica el nombre de la tabla a `data_user:`

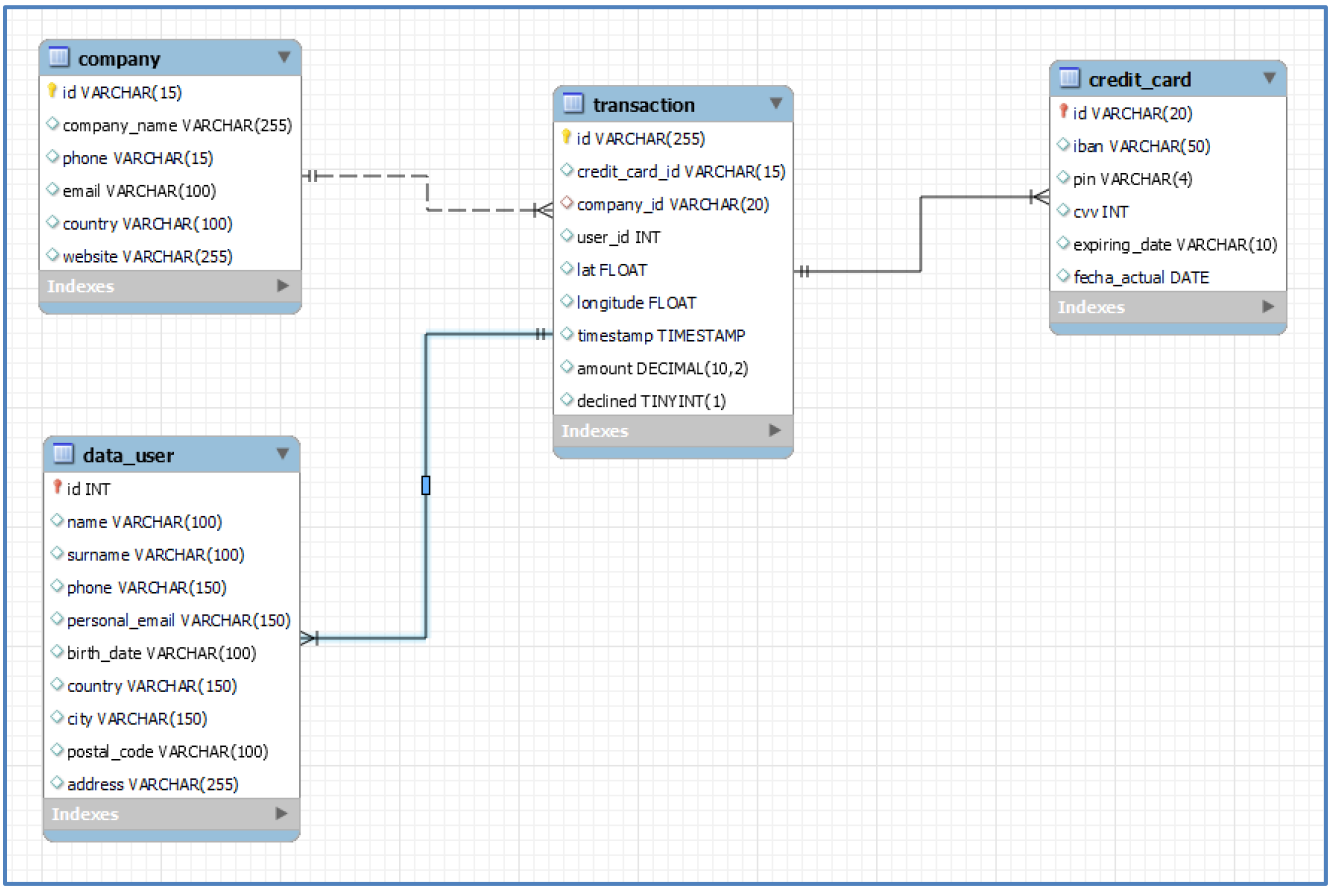
```
732 •   rename table user to data_user;
```

Paso 8: En tabla `data_user`, se modifica el nombre del campo `email` a `personal_email`:

```
736 •   alter table data_user
737     change column email personal_email varchar(150);
```

Como consecuencia de aplicar todas los pasos obtenemos la misma tabla como la mostrada en el enunciado; con lo que se le podrá orientar al compañero del equipo los comandos que tuvo que ejecutar paso a paso, tanto para volver a ejecutarlos como para poder revertirlos, de ser necesario.

Diagrama final, luego de ejecutar todos los pasos:



Ejercicio 3.2

La empresa también solicita 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
- Asegurarse de incluir información relevante de ambas tablas y utilizar alias para cambiar de nombre columnas según sea necesario.

Mostrar los resultados de la vista, ordenar los resultados de forma descendente en función de la variable ID de transacción.

Hemos creado una vista o **`view`** según los datos solicitados, adicionándole el monto de la transacción.

El script ejecutado y su respectivo resultado se muestran a continuación:

```
741   # Ejercicio 3.2 : Crear una vista llamada "InformeTecnico" conteniendo la siguiente información:
742   /*
743   ID de la transacción; nombre del usuario/a, apellido del usuario/a; IBAN de la tarjeta de credito usada;
744   nombre de la compañía de la transacción realizada.
745   Mostrar resultados de la vista, ordenar los resultados de forma descendente en función de la
746   variable ID de transacción.
747   */
748
749   create or replace view InformeTecnico as
750   select transaction.id as id_transaccion, data_user.name as nombre_usuario, data_user.surname as apellido_usuario,
751   credit_card.iban as iban_tarjeta_credito, transaction.amount as monto_transaccion, company.company_name as nombre_compañia
752   from transaction
753   join company on company.id = transaction.company_id
754   join credit_card on credit_card.id = transaction.credit_card_id
755   join data_user on data_user.id = transaction.user_id
756   order by transaction.id desc;
757
758   select * from InformeTecnico;
```

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

id_transaccion	nombre_usuario	apellido_usuario	iban_tarjeta_credito	monto_transaccion	nombre_compañia
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	480.13	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E4646B	Molly	Gilliam	SE2813123487163628531121	219.83	Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	42.32	Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	200.72	Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	78.29	Neque Tellus Imperdiet Corp.
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	Hakeem	Alford	MD1234119525145401270486	335.56	Nunc Interdum Incorporated
FBD7E0D6-BA6B-F5BC-0CA9-EA4B8760100C	Hedwig	Gilbert	MU413233344453434254134478885	207.09	Mauris Id Inc.

InformeTecnico 56

Output

Action Output

#	Time	Action	Message
381	14:28:09	create or replace view InformeTecnico as select transaction.id as id_transaccion, data_user.name a...	0 row(s) affected
382	14:28:11	select * from InformeTecnico LIMIT 0, 50000	586 row(s) returned